

(19) 日本国特許庁 (JP)

# (12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-290724

(P 2001-290724 A)

(43) 公開日 平成13年10月19日 (2001. 10. 19)

(51) Int. Cl.	識別記号	F I	マークコード (参考)
G06F 13/00	357	G06F 13/00	357 A
	351		351 B
13/38	350	13/38	350
15/00	310	15/00	310 D
H04L 12/28		H04L 11/00	310 Z

審査請求「有」請求項の数32：O L (全24頁) 最終頁に続く

(21) 出願番号 特願2000-371402 (P 2000-371402)

(22) 出願日 平成12年12月6日 (2000. 12. 6)

(31) 優先権主張番号 09/546397 (P)

(32) 優先日 平成12年4月10日 (2000. 4. 10)

(33) 優先権主張国 米国(US)

(71) 出願人 000004237 日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 ビシャル・シャー、アーヴィング  
アメリカ合衆国、カリフォルニア 95134  
サンノゼ、110プロブルスアベニュー、エバ  
ヌ・イー・シー・ユー・エス・エー・イン  
ク内

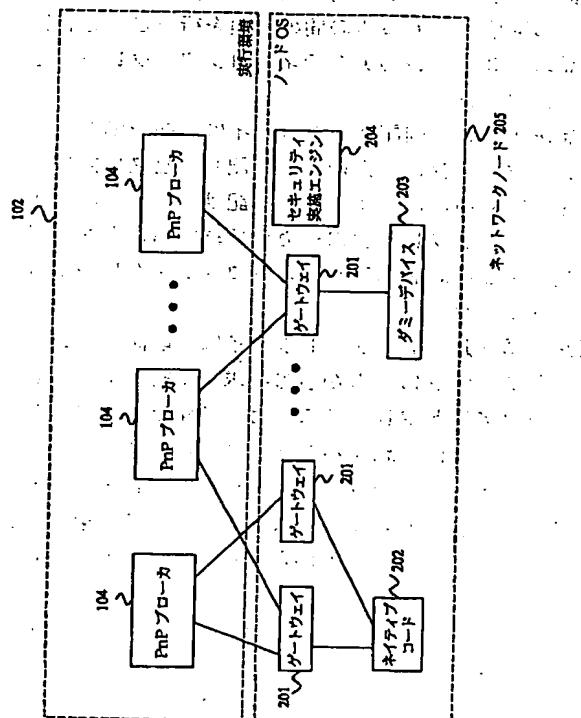
(74) 代理人 100097157 弁理士 桂木 雄二  
弁理士 桂木 雄二

(54) 【発明の名称】 プラグアンドプレイ機能を有するフレームワークおよびその再構成方法

## (57) 【要約】

【課題】 相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする。

【解決手段】 本発明のアクティブコンフィグレーションフレームワークシステムは、サービスとサービスユーザを相互接続する。このフレームワークは、プラグアンドプレイ(PnP)プローカ104を有する。サービスユーザは、PnPプローカを用いて、ネットワーキングサービスを発見し、利用し、ネットワーキングサービスと通信する。サービスユーザは、サービスを提供するデバイスによって使用される通信プロトコルとは独立に、PnPプローカのインターフェースを通じてサービスと通信する。また、サービスをPnPプローカに登録するゲートウェイ201が提供される。



1

## 【特許請求の範囲】

【請求項1】 (a) 少なくとも1つのサービスおよび少なくとも1つのサービスユーザと、  
 (b) 第1のプラグアンドプレイプローカとを有するアクティブコンフィグレーションフレームワークにおいて、前記少なくとも1つのサービスユーザは、前記第1のプラグアンドプレイプローカを用いて、前記サービスを利用することを特徴とするアクティブコンフィグレーションフレームワーク。

【請求項2】 前記サービスユーザは、前記サービスを提供するデバイスによって使用される通信プロトコルとは独立に、前記第1のプラグアンドプレイプローカを通じて前記サービスと通信することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項3】 前記サービスは、ゲートウェイおよびデバイスクラスタを有し、前記ゲートウェイは、前記デバイスクラスタを前記第1のプラグアンドプレイプローカに接続し、前記第1のプラグアンドプレイプローカは、前記ゲートウェイを通じて、前記デバイスクラスタと通信することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項4】 前記第1のプラグアンドプレイプローカは、第1の通信プロトコルを用いて前記ゲートウェイに接続され、

前記ゲートウェイは、異なる第2の通信プロトコルを用いて前記デバイスクラスタに接続され、

前記ゲートウェイは、前記第1の通信プロトコルと前記第2の通信プロトコルの間の変換を行うことを特徴とする請求項3に記載のアクティブコンフィグレーションフレームワーク。

【請求項5】 前記ゲートウェイは、前記第1のプラグアンドプレイプローカ内の、前記サービスに対応する属性を設定することによって、前記サービスを前記第1のプラグアンドプレイプローカに登録することを特徴とする請求項3に記載のアクティブコンフィグレーションフレームワーク。

【請求項6】 前記ゲートウェイは、前記第1のプラグアンドプレイプローカからの要求に応答して、前記第1のプラグアンドプレイプローカで前記サービスのステータスを周期的に更新することを特徴とする請求項3に記載のアクティブコンフィグレーションフレームワーク。

【請求項7】 前記サービスユーザは、前記第1のプラグアンドプレイプローカから離れたリモートサービスユーザであり、

前記リモートサービスユーザは、インターネット通信プロトコルを通じて前記第1のプラグアンドプレイプローカと通信することを特徴とする請求項1に記載のアクテ

10

ィブコンフィグレーションフレームワーク。

【請求項8】 前記アクティブコンフィグレーションフレームワークは、第2のプラグアンドプレイプローカをさらに有し、

前記サービスユーザは、前記第2のプラグアンドプレイプローカと通信し、

前記第2のプラグアンドプレイプローカは、前記第1のプラグアンドプレイプローカと通信して、前記少なくとも1つのサービスユーザが前記サービスを利用することを可能にすることを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

20

【請求項9】 前記アクティブコンフィグレーションフレームワークは、ディレクトリエージェントをさらに有し、

前記第1のプラグアンドプレイプローカは、サービスロケーションプロトコルを用いて、前記サービスを前記ディレクトリエージェントに登録することを特徴とする請求項8に記載のアクティブコンフィグレーションフレームワーク。

20

【請求項10】 前記第1のプラグアンドプレイプローカは、サービスレジストリを有し、

前記サービスレジストリは、前記サービスおよび他のネットワークサービスに関する情報を含み、前記少なくとも1つのサービスユーザは、前記サービスレジストリを用いて、前記サービスおよび前記他のネットワークサービスを発見することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

30

【請求項11】 前記アクティブコンフィグレーションフレームワークは、第2のプラグアンドプレイプローカをさらに有し、

前記第2のプラグアンドプレイプローカは、前記第1のプラグアンドプレイプローカに問合せを送ることによって、前記サービスを発見することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

40

【請求項12】 前記アクティブコンフィグレーションフレームワークは、第2のプラグアンドプレイプローカおよびディレクトリエージェントをさらに有し、

(a) 前記少なくとも1つのサービスユーザは、要求されるサービスの記述を前記第2のプラグアンドプレイプローカに送り、

(b) 前記第2のプラグアンドプレイプローカは、前記ディレクトリエージェントと通信し、

(c) 前記ディレクトリエージェントは、前記要求されるサービスの記述に一致するサービスのアドレスを前記第2のプラグアンドプレイプローカに返し、

50

(d) 前記第2のプラグアンドプレイプローカは、前記アドレスを用いて、前記サービスユーザと前記サービスの間にセッションを確立することを特徴とする請求項1

に記載のアクティブコンフィグレーションフレームワーク。

【請求項13】 (1) 前記サービスは、ゲートウェイおよびデバイスクラスタを有し、

(2) 前記ゲートウェイは、前記デバイスクラスタを前記第1のプラグアンドプレイプローカに接続し、

(3) 前記第1のプラグアンドプレイプローカは、前記ゲートウェイを通じて、前記デバイスクラスタと通信し、

(4) 前記ゲートウェイは、前記第1のプラグアンドプレイプローカ内の、前記サービスに対応する属性を設定することによって、前記サービスを前記第1のプラグアンドプレイプローカに登録し、

(5) 前記第1のプラグアンドプレイプローカは、前記サービスを前記ディレクトリエージェントに登録することを特徴とする請求項10に記載のアクティブコンフィグレーションフレームワーク。

【請求項14】 前記第1のプラグアンドプレイプローカは、

(a) 前記サービスユーザと通信するプラグアンドプレイプローカインタフェースと、

(b) 前記第1のプラグアンドプレイプローカと、前記フレームワーク内の他のプラグアンドプレイプローカとの間の通信のためのプラグアンドプレイプローカ間プロトコルと、

(c) 前記サービスに関する情報を格納する少なくとも1つのサービスレコードを有するサービスレジストリと、

(d) 前記サービスのネットワークアドレスを探索し前記サービスのアベイラビリティステータスを取得するサービスディスカバリ／アベイラビリティエージェントと、

(e) 前記サービスと前記サービスユーザの間の通信セッションを確立するサービスセッション管理エージェントと、

(f) 前記サービスのアドレスを探索するサービスロケーションプロトコルユーザエージェントと、

(g) 前記サービスを公表するサービスロケーションプロトコルサービスエージェントとを有することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項15】 前記少なくとも1つのサービスレコードは、前記サービスのタイプおよび複数のサービス属性レコードを格納するサービス識別フィールドを有し、前記サービス属性レコードは、前記サービス属性のタイプを格納する属性識別フィールドと、前記サービス属性の値を格納する属性値フィールドと、前記サービスのアクセス制御情報を格納するアクセス制御フィールドとを有することを特徴とする請求項14に記載のアクティブコンフィグレーションフレームワーク。

【請求項16】 (a) 少なくとも1つのサービスと、  
(b) サービスエージェントと、  
(c) ディレクトリエージェントとを有するアクティブコンフィグレーションフレームワークにおいて、前記サービスエージェントは、前記少なくとも1つのサービスのタイプおよび属性を前記ディレクトリエージェントに登録し、

前記ディレクトリエージェントは、ユーザアプリケーションによって要求されるサービスのタイプおよび属性を、前記少なくとも1つのサービスのタイプおよび属性と照合して、前記少なくとも1つのサービスのアドレスを返すことを特徴とするアクティブコンフィグレーションフレームワーク。

【請求項17】 前記アクティブコンフィグレーションフレームワークは、ユーザエージェントをさらに有し、前記ユーザエージェントは、前記要求されるサービスのタイプおよび属性を前記ディレクトリエージェントに送信し、

前記ディレクトリエージェントは、前記少なくとも1つのサービスのアドレスを前記ユーザエージェントに返すことを特徴とする請求項16に記載のアクティブコンフィグレーションフレームワーク。

【請求項18】 前記少なくとも1つのサービスは、ゲートウェイおよびデバイスクラスタを有し、前記ゲートウェイは、前記デバイスクラスタを前記サービスエージェントに接続し、前記サービスエージェントは、前記ゲートウェイを通じて、前記デバイスクラスタと通信することを特徴とする請求項16に記載のアクティブコンフィグレーションフレームワーク。

【請求項19】 前記アクティブコンフィグレーションフレームワークは、実行環境をさらに有し、前記アプリケーションは、前記実行環境で動作することを特徴とする請求項16に記載のアクティブコンフィグレーションフレームワーク。

【請求項20】 前記実行環境はJava仮想マシンであることを特徴とする請求項19に記載のアクティブコンフィグレーションフレームワーク。

【請求項21】 前記実行環境はWindows™ベースのマシンであることを特徴とする請求項19に記載のアクティブコンフィグレーションフレームワーク。

【請求項22】 ネットワークデバイスの追加に応答してフレームワークの自動再構成を行う方法において、前記ネットワークデバイスは、少なくとも1つのサービスを提供し、

前記フレームワークは、サービスエージェントおよびディレクトリエージェントを有し、

前記方法は、

(a) 前記サービスエージェントが、提供されるサービスのタイプおよび属性を前記ディレクトリエージェント

に登録するステップと、

(b) 前記ディレクトリエージェントが、ユーザアプリケーションによって要求されるタイプおよび属性を、前記提供されるサービスのタイプおよび属性と照合して、前記提供されるサービスのアドレスを返すステップと、を有することを特徴とする、ネットワークデバイスの追加に応答してフレームワークの自動再構成を行う方法。

【請求項23】 前記フレームワークは、ユーザエージェントをさらに有し、

前記ユーザエージェントは、前記要求されるサービスのタイプおよび属性を前記ディレクトリエージェントに送信し、

前記ディレクトリエージェントは、前記提供されるサービスのアドレスを前記ユーザエージェントに返すことを特徴とする請求項22に記載の方法。

【請求項24】 前記ステップ(a)は、

(1) 前記ゲートウェイが、前記ネットワークデバイスのアベイラビリティをチェックするステップと、

(2) 前記ゲートウェイが、前記提供されるサービスのタイプおよび属性を前記サービスエージェントに登録するステップと、

(3) 前記サービスエージェントが、前記提供されるサービスのタイプおよび属性を前記ディレクトリエージェントに登録するステップと、

を有することを特徴とする請求項22に記載の方法。

【請求項25】 (c) セッション管理エージェントが、前記アドレスを用いて、前記ユーザアプリケーションと前記ネットワークデバイスの間の通信セッションを確立するステップをさらに有することを特徴とする請求項22に記載の方法。

【請求項26】 フレームワークに挿入されたデバイスによって実行されるサービスのための通信インターフェースを生成する方法において、

前記インターフェースは、ユーザアプリケーションによって、前記サービスを利用するためには使用され、前記方法は、

(a) XML文書、文書型定義、またはスタイルシートのうちの少なくとも1つを含む、前記デバイスによって提供されるサービスのサービス記述スキーマを生成するステップと、

(b) XMLバーサを用いて、前記サービス記述スキーマから、前記サービスのための通信インターフェースを生成するステップと、

を有することを特徴とする、フレームワークに挿入されたデバイスによって実行されるサービスのための通信インターフェースを生成する方法。

【請求項27】 前記サービス記述スキーマは、前記サービスのためのデータおよび制御フロー仕様を含むことを特徴とする請求項26に記載の方法。

【請求項28】 前記ステップ(b)において、前記デ

バイスによって生成されるサービス記述スキーマは、ユーザインターフェースウィジェットのセットにマッピングされ、

ユーザは、実行のために前記デバイスにコマンドを送るユーザインターフェースウィジェットをクリックすることによって、前記デバイスの状態を変えることができることを特徴とする請求項26に記載の方法。

【請求項29】 前記ステップ(b)において、前記デバイスによって生成されるサービス記述スキーマは、前記サービスのための通信インターフェースを表すオブジェクトの階層にマッピングされ、

前記サービスは、前記オブジェクトの階層を用いて、前記ユーザアプリケーションによって利用されることを特徴とする請求項26に記載の方法。

【請求項30】 前記オブジェクトの階層の構造は、前記ユーザアプリケーションが前記デバイスを利用する時点での前記デバイスの機能に基づいて決定されることを特徴とする請求項29に記載の方法。

【請求項31】 前記XMLバーサは、XML対応ウェブブラウザであることを特徴とする請求項30に記載の方法。

【請求項32】 フレームワークに挿入されたデバイスを制御する方法において、該方法は、

(a) 前記デバイスが、XML文書、文書型定義、またはスタイルシートのうちの少なくとも1つを含む、前記デバイスのデバイス記述スキーマを生成するステップと、

(b) 前記ユーザアプリケーションが、前記デバイス記述スキーマを編集し、編集されたデバイス記述スキーマを前記デバイスに反映するステップとを有し、

前記デバイスは、前記編集されたデバイス記述スキーマに従って状態を変更することを特徴とする、フレームワークに挿入されたデバイスを制御する方法。

#### 【発明の詳細な説明】

##### 【0001】

【発明の属する技術分野】 本発明は、相異なる通信プロトコルを有する複数のネットワークを单一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用すること可能にする方法および装置に関する。また、本発明は、利用可能な機能の変更およびユーザ相互作用に応答してネットワークノードの動的な適応および再構成を達成する方法に関する。また、本発明は、アプリケーション、サービス、およびデバイスが、自己の能力を記述し、それを他のアプリケーション、サービス、およびデバイスに公表する方法に関する。さらに、本発明は、物理的に相異なるデバイスが、接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供して、ネットワークプラグアンドプレイを達成すること

50 を可能にする、プラットフォーム独立でトランスポート

独立なプロトコルを提供する方法に関する。

【0002】

【従来の技術】世界的なネットワーキング基盤の出現により、分散サービスおよび分散アプリケーションの大規模な配備が一般的になっている。現在および将来のネットワークでは、時間に敏感な情報を全世界に発信し、世界的な取引を電子的に仲介するサービスがさらに配備されることが期待される。このような分散サービスが普遍的になる前に、このようなサービスの開発、デバッグ、配備および発展を簡単にする新たな機構が必要とされる。このような機構は、基礎になるプロトコルやインターフェースにかかわらずに、必要なサービスを発見し、そのサービスの能力を使用することが必要となる。

【0003】家庭で使用可能なもののような典型的な分散ネットワークシステムは、相異なるさまざまな機器を相互接続する。さまざまな家庭用機器（ホームデバイス）は本質的にさまざまなインターフェースおよびプロトコルを利用するため、単一の共通の通信プロトコルを用いてこれらのすべてのデバイスを相互接続することはほとんど不可能である。場合によっては、ほとんどの家庭に既に設置されている配線を使用する（例えば、既存の電源線をX-10デバイスとの通信に使用する）ことは便利であるが、このような既存の配線を使用するネットワークの伝送速度が制限される。例えば、白熱電球やホームセキュリティシステムのようなX-10デバイスは、毎秒数ビットのデータレートで動作する。他方、デジタルTV、カメラ、あるいはVCRのようなデバイスは、ずっと高い伝送データレートを必要とする。このようなデバイスは通常、IEEE1394アキテクチャプロトコルを用いて相互接続され、データ転送レートは400Mbpsに達する。

【0004】これに対して、現在開発されているさまざまな家庭用自動化アプリケーションは、さまざまな家庭用機器の有効な相互作用を必要とする。したがって、家庭に設置されたすべての相異なるデバイスおよびサービスの有効な相互作用を可能にするには、相異なるさまざまなネットワークを、統一するフレームワークに統合して、相異なるネットワーク内に位置するさまざまなネットワークエンティティが互いを発見し相互作用する機構を提供しなければならない。

【0005】また、さまざまなデバイスインターフェースおよびネットワークプロトコルについて、低レベルのプロトコル固有の詳細をアプリケーションが扱うことを必要とせずに、リモートサービスを利用する手段をユーザアプリケーションに提供することが望ましい。

【0006】真のプラグアンドプレイ能力を達成するには、フレームワークは、新たに追加されたデバイスによって提供されるサービスを自動的に発見する機能を提供しなければならない。このようなフレームワーク内では、利用可能なネットワークサービスおよびそのネット

ワーク位置に関する記述は、すべてのユーザアプリケーションにとって容易に利用可能であるようにしなければならない。さらに、ユーザアプリケーションは、遭遇する可能性のある特定のデバイスあるいはサービスとの通信のために、自己の通信インターフェースを自動的に再構成することができなければならない。さらに、フレームワークは、ネットワークサービスへのアクセスを制御するために、ユーザの認可および認証を実行するセキュリティモデルを含まなければならない。

10 【0007】Jini<sup>TM</sup>は、ネットワークプラグアンドプレイを達成する既存の候補のうちの1つである（Jiniは、サンマイクロシステムズ社（Sun Microsystems, Inc.）の商標である）。Jiniは、分散システムの構築および配備を容易にするアプリケーションプログラミングインターフェース（API）およびランタイム規約のセットである。Jiniは、分散システムの、共通しているが相異なる部分を処理する「配管」（plumbing）を提供する。Jiniは、プログラミングモデルおよびラン

20 タイムインフラストラクチャからなる。リース、分散イベント、および分散トランザクションをサポートするAPIおよび規約を定義することによって、プログラミングモデルは、基礎となるネットワークの信頼性が低くても、信頼性の高い分散システムを開発者が構築するのを助ける。ランタイムインフラストラクチャは、ネットワークプロトコルとそれを実装するAPIとからなり、ネットワーク上のデバイスおよびサービスの追加、探索、アクセス、および削除を容易にする。

【0008】Jiniの使用は、基礎となるネットワーク技術について知らなくてもサービスを発見する容易な方法を提供する。他の既知のインフラストラクチャに比べてJiniが改良されている点は、ユーザアプリケーションが実際にサービスを探索し、そのサービスをサポートするホストを他者に発見させることができることである。しかし、主な未解決の問題点は、「ユーザアプリケーションはどのようにして自動的にクライアントマシン上のサービスを使用するか」である。クライアントユーザアプリケーションにサービスインターフェースしか設けられていない場合、クライアントが、このサービスを理解することができるコードを有していないなければならない。

【0009】例えば、ユーザが、自己のアプリケーションに、カメラサービスのためのネットワークをブラウズさせたい状況を考える。アプリケーションがこのようなサービスをネットワーク上に見つけた場合、ユーザは、それをクリックして、ユーザ自身のアプリケーションでそれを使用することができる。ユーザが、このようなサービスを実行するためのコードを必要とする場合、このようなコードは、ユーザのマシン上に自動的にダウンロードされなければならない。しかし、このアプローチに関連する主な問題点は、ユーザアプリケーションがこの

のようなサービスをどのようにして利用するかである。ユーザアプリケーションは、拡張可能なインターフェースを有する場合、リモートサービスインターフェースに問い合わせてその固有の特性を取得し、その利用を可能にすることができる。しかし、このような拡張可能インターフェースの定義は、Jini<sup>TM</sup>仕様の範囲内ではなく、Jini<sup>TM</sup>サービスを使用するユーザアプリケーションに委ねられている。

【0010】もう1つの問題点は、カメラサービスのセマンティクス（意味論）を定義するエンティティの識別である。例えば、キャノン、ミノルタ、およびニコンを含む主要なカメラメーカーがそれぞれ固有のインターフェースを提供する場合、カメラサービスには相互運用性がなくなり、ユーザは、3つの異なるすべてのカメラサービスをコードの形で適切に提供しなければならないことになる。また、もしコダックがJini<sup>TM</sup>カメラを市販することに決めた場合、ユーザは、コダックのインターフェースの解析を含むようにそのコードを手動（マニュアル）で変更しなければならない。このような拡張可能インターフェースは、ネットワークトラフィックと、クライアントコードのサイズを非常に増大させてしまう。すべての種類の市場固有のグループがまとまってインターフェースを指定することが好ましく、多くの場合にはこれが必要となるであろうが、これはすぐには、また効率的には実現しそうになく、特に、家庭で使用されるすべての種類のデバイスに対しては、きわめて非現実的である。

【0011】既知のネットワークで依然として解決されていないもう1つの制限は、Jini<sup>TM</sup>ベースのアプリケーションがどのようにして非Jini<sup>TM</sup>ベースのオペレーティングシステム上にあるデータにアクセスすることができるかに関するものである。すなわち、ユーザは、例えば、Jini<sup>TM</sup>に対するマイクロソフトのサポートなしで、どのようにしてMS-Word<sup>TM</sup>からJini<sup>TM</sup>プリントにアクセスすることができるだろうか。Jini<sup>TM</sup>プリントを使用したいJava<sup>TM</sup>アプリケーションですら、アクセスのためのオペレーティングシステムに依拠しなければならない。

【0012】例えば、ユーザプログラムが印刷メソッドへのアクセスを要求するとき、以下のようなコードフレグメントが使用される。

```
object o=Lookup(GeneralPrintService);
GeneralPrintService pservice=(GeneralPrintService)
o;
pservice.print();
```

【0013】しかし、このコードが書かれるときに、プログラマは、GeneralPrintServiceという名前のインターフェースがコード実行時に本当に存在するかどうかを知らない。さらに、プログラマが複数のデバイスでイメージを印刷したい場合、複数のインターフェースを予想し、それに応じて印刷を行う必要がある。

【0014】Jini<sup>TM</sup>は、ユーザインターフェース（UI）コンポーネントがサービスの属性リストにエントリとして付加されるように使用することができるため、サービスについて要求されるクライアントの知識は最小限である。クライアントは、単に、UIがアプレットと同様の属性によって表現されることを知っているだけでよい。UIは、JavaBeansとして提供可能であるため、内省(introspection)・反省(reflection)により、サービスは、クライアントに対して、クライアントがそのサービスを使用することができる手段を提供することが可能である。これは、本質的に、クライアントとサーバの間のメッセージ交換であり、手動の設定を必要とする。しかし、UIコンポーネントを付加することは、人間でないクライアント（生成・使用されるマシン）がデバイスを使用するのを妨げる可能性がある。また、デバイスの個数が増大して、クライアントプログラムが複雑になると、もう1つの問題点が生じる。その場合、クライアントは、返されたリストのどのオブジェクトが真のターゲットであるかを決定しなければならない。さらに、あらゆるJini<sup>TM</sup>のデバイスあるいはアプリケーションは、製造時において通信するために必要としたすべてのデバイスの（あるいは少なくとも、「デバイスのすべてのタイプの」）すべてのインターフェースを必要とする。その結果、Jini<sup>TM</sup>は、相互運用性の問題点を実際には解決していない。

【0015】もう1つの既存のネットワーキングシステムであるHome APIは、アプリケーションがさまざまなホームデバイスを発見し利用することを可能にする、オブジェクト指向のソフトウェアサービスおよびアプリケーションプログラミングインターフェース（API）のセットである。このシステムは、アプリケーションが使用するデバイスおよびネットワークのプロトコル固有の詳細に、ユーザアプリケーションが関与しないように設計された。Home APIは、Home APIインターフェースの使用を通じてサービスを利用するユーザアプリケーションを開発するのを容易にする分散サービスを表現するソフトウェアオブジェクトのセットを提供する。

【0016】しかし、Home APIは、新たなデバイスの発見の問題点を解決していないため、所望のプラグアンドプレイネットワーキング能力を提供しない。Home APIのもう1つの欠点は、すべてのネットワークデバイスが標準の通信インターフェースを提供することを要求することであり、これは今日の市場では実現不可能である。

【0017】HomePNA、HomeRF、Blue tooth、PIANO、X-10、CEBus、IEEE1394、USB、HDMIなどの他の発展段階のネットワーク構想は、単に、ネットワーク接続に関するさまざまなプロトコルを提供しているだけ

ある。これらの通信プロトコルは当業者に周知である。しかし、現在のところ、競合するさまざまなネットワーク技術およびプロトコルの相互運用性の問題点を解決し、ネットワークプラグアンドプレイを達成する統一フレームワークシステムは存在しない。

#### 【0018】

【発明が解決しようとする課題】したがって、さまざまなタイプのネットワークデバイスを相互接続し、デバイスの発見および利用のための一様な機構を提供する統一フレームワークに対して、強く幅広い受容が認識されており、このようなものがあることは非常に有利である。このような統一フレームワークは、ユーザアプリケーションが、アプリケーションーサービス間通信プロトコル、所望のサービスを提供するデバイスに接続されたホストのIPアドレスの発見、およびさらには、ネットワークサービスのユーザインタフェースの詳細のような低レベルの詳細を組み込むことを不要にするとともに、上記のその他の要求を満たす。

#### 【0019】

【課題を解決するための手段】したがって、本発明の1つの目的は、既知のネットワーキングシステムの上記の制限を克服し、さまざまな機器を相互接続する相異なる通信プロトコルを有する複数のネットワークを单一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする方法および装置を提供することである。

【0020】具体的には、本発明の目的は、利用可能な機能の変更およびユーザ相互作用に応答してネットワークノードの動的な適応および再構成を達成する方法および装置を提供することである。

【0021】本発明のもう1つの目的は、アプリケーション、サービス、およびデバイスが、自己の能力および通信インタフェースを記述し、それを他のアプリケーション、サービス、およびデバイスに公表する方法を提供する。

【0022】本発明のさらにもう1つの目的は、物理的に相異なるデバイスが、接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供して、ネットワークプラグアンドプレイを達成することを可能にする、プラットフォーム独立でトランスポート独立なプロトコルを提供する方法および装置を提供することである。

【0023】上記の目的を実現し、本発明の効果を達成するため、アクティブコンフィグレーションフレームワークシステムが提供される。本発明のフレームワークは、サービスとサービスユーザを相互接続する。また、このフレームワークは、プラグアンドプレイプロローカを有する。本発明のフレームワークでは、サービスユーザは、プラグアンドプレイプロローカを用いて、ネットワー

10

キングサービスを発見し、利用し、ネットワーキングサービスと通信する。

【0024】本発明のフレームワークでは、サービスユーザーは、サービスを提供するデバイスによって使用される通信プロトコルとは独立に、プラグアンドプレイプロローカのインターフェースを通じてサービスと通信する。

【0025】また、本発明によれば、サービスをプラグアンドプレイプロローカに登録するゲートウェイが提供される。

10

【0026】また、本発明によれば、サービスエージェント、ユーザエージェント、およびディレクトリエージェントを有するアクティブコンフィグレーションフレームワークが提供される。本発明のフレームワークでは、サービスエージェントはサービスをディレクトリエージェントに登録する。その後、ユーザエージェントは、要求するサービスの記述をディレクトリエージェントに通信し、ディレクトリエージェントは、適合するサービスのロケーションを帰す。

10

【0027】本発明のフレームワークでは、サービスは、デバイスクラスタおよびゲートウェイからなることが可能である。サービスエージェントは、このゲートウェイを通じてデバイスクラスタと通信する。

20

【0028】また、本発明によれば、新たなネットワークデバイスの追加に応答して、フレームワークの自動的再構成を行う方法が提供される。本発明の方法によれば、サービスエージェントは、サービスをディレクトリエージェントに登録する。その後、ユーザエージェントは、要求するサービスの記述をディレクトリエージェントに通信する。最後に、ディレクトリエージェントは、要求されたサービスの記述を、提供するサービスの記述と照合して適合したサービスのアドレスをユーザエージェントに返す。

20

【0029】また、本発明によれば、フレームワークに追加される新たなネットワークデバイスによって実行されるサービスに対する通信インタフェースを生成する方法が提供される。本発明の方法によれば、デバイスは、その通信インタフェースを記述するスキーマを生成する。このスキーマは、XML (eXtensible Markup Language) 文書の形式とすることが可能である。その場合、ユーザアプリケーションは、XML言語バーサを用いて、通信インタフェースを生成する。

40

【0030】また、本発明によれば、ネットワークデバイスを制御する方法が提供される。本発明の方法によれば、ユーザアプリケーションは、デバイス記述スキーマを編集し、編集されたデバイス記述スキーマをデバイスに返す。これに応答して、デバイスは、編集されたデバイス記述スキーマに従って状態を変更する。

#### 【0031】

【発明の実施の形態】すべてのタイプの分散デバイスおよび分散サービスにわたりネットワークプラグアンドプ

レイを達成するためには、統一フレームワークが必要である。ここで使用される「フレームワーク」あるいは「サブストレート」という用語は、1つ以上のネットワークを統合するシステムを意味する。しかし、多くの場合、「フレームワーク」、「サブストレート」、および「ネットワーク」という用語は区別なく用いられる。

【0032】本発明のフレームワークを使用することにより、ネットワーク内の相異なる物理デバイスの個数にかかわらず、ネットワーキングエンティティの相互作用のための複雑な機構は一度だけ実装すればよい。さらに、これらの機構は、デバイス固有のゲートウェイを通じて管理プロローカと相互作用するユーザアプリケーションから隠蔽される。本発明のフレームワークは、S L P、H T T P、X M L、J a v a R M I、およびI I O Pのような周知のインターネット関連プロトコルに基づいており、ウェブサーバへのアドオンとして実装可能であるため、配備および利用が容易である。さらに、本発明のフレームワークでは、サービス発見の全プロセスは、ユーザアプリケーションにとって透過的（トランスペアレント）であり、物理接続（コネクション）に対する区別がない。例えば、ネットワークに接続されるプラグアンドプレイデバイスはT C P／I Pベースである必要はない。サービス利用プロセスもまたトランスペアレントであり、対象となるさまざまなデバイスの能力と、ユーザの認証および認可に基づく。互換性のないデバイスを接続するときには、トンネリングや完全なデータ変換が必要となることもある。

【0033】本発明は、デバイス独立な情報交換を容易にするためのアクティブコンフィグレーションフレームワークを提供する。複数のデバイスが接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供するには、相異なるプロトコルが要求される。これらのプロトコルは、プラットフォーム独立であり、その形式や機能にかかわらず、他のデバイスに接続する必要のある任意のデバイスに組み込むことができる。これらのプロトコルはトランスポート独立であるため、任意のデバイスが、他の任意のデバイスに接続し、情報交換することが可能となる。

【0034】本発明のフレームワークでは、アクティブネットワークの場合のように、ユーザ定義計算がネットワーク内に配置される。しかし、アクティブネットワークとは異なり、現在のインターネットアーキテクチャのすべてのルーティングおよびフォワーディングのセマンティクスは、計算環境をアプリケーション層に制限することにより保持される。このフレームワークを利用するネットワークデバイスおよびサービスは、ネットワークインフラストラクチャに深く組み込まれる場合でも、実際には、エンドシステム上のユーザアプリケーションによって作成され、設定され、動的に制御される。このようなアプリケーション定義エンティティは、ネットワー-

ク内の任意のノードで実行され、個々のパケットは、ネットワークを通じて伝搬するとともに任意のアクションを実行するようにプログラムされることが可能である。

【0035】【アクティブコンフィグレーションフレームワークのコンポーネント】完全なアクティブコンフィグレーションアーキテクチャは、ダミーデバイス、ゲートウェイデバイス、および、ネットワークに直接的または間接的に接続されたP n Pプロローカからなる。アクティブコンフィグレーションフレームワークのアーキテクチャを図1に示す。図1に示すように、ネットワークには1個以上の実行環境102がある。実行環境102は、例えば、インターネットのようなネットワーク101を用いて相互接続される。実行環境102は、ある意味で、ネットワークノードとして作用する。実行環境102は、当業者に周知のJ a v a仮想マシンやW i n d o w s<sup>TM</sup>ベースのマシン内にあることが可能である。実行環境102は、ゲートウェイ（図示せず）を通じて非インテリジェント（すなわち、ダミー）デバイスに接続された1個以上のP n Pプロローカ104を有する。ユーザアプリケーション103は、P n Pプロローカ104を通じて、デバイスを発見し、利用し、デバイスと通信する。以下で、本発明のアクティブコンフィグレーションフレームワークシステムのさまざまなコンポーネントについて詳細に説明する。

【0036】【ダミーデバイス】ダミーデバイスは、T C P／I Pプロトコルスタック、もしくはJ a v a仮想マシン、またはその両方のないデバイスである。このようなデバイスは、ネットワークから直接にアクセス可能ではなく、媒介物としてゲートウェイを必要とする。このようなデバイスは、ネットワークにインストールされる前に、特定のコンフィグレーション選択を行う必要がある。例えば、それぞれのX-10デバイス（白熱電球など）には、電源線ネットワークに接続する前に手動でハウスコードおよびデバイスコードを割り当てなければならない。さらに、このようなデバイスは、提示するサービスのためのセキュリティやアクセス制御を実施することはない。このようなデバイスのほとんどは、メモリや処理能力を有しておらず、ネットワークの残りの部分に依然として接続する必要のあるレガシーデバイスとして特徴づけることができる。

【0037】【ゲートウェイデバイス】通常のネットワークデバイスは、単一のネットワークタイプ（例えば、I PまたはI P X）により、ピアツーピア方式で相互作用することができる。これは、あるネットワークタイプのデバイスは、同じネットワークに物理的に接続された、同じネットワークタイプのデバイスとのみ通信することができることを意味する。第1のデバイスが、その第1のデバイスによってサポートされていないネットワーク上にある第2のデバイス、あるいは、第1のデバイスによってサポートされていないプロトコルを実行して

いる第2のデバイスに接続することを必要とする場合に問題が生じる。この問題を解決するには2つのアプローチがある。

【0038】第1のアプローチは、複数の通信スタックを単一のネットワークデバイスにロードすることである。これにより、デバイスは、適当な通信スタックを有する限り、任意のタイプのネットワークデバイスと相互作用することが可能となる。基礎となるトランスポートは異なるが、デバイスがプロトコルスタック全体を理解する場合には、このアプローチはうまくいく。

【0039】複数の相異なるデバイスを接続する問題を解決するため、本発明のフレームワークシステムはゲートウェイを使用する。ゲートウェイは、ネットワーク内に不可視的に存在し、あるネットワークタイプを別のネットワークタイプに変換するデバイスである。このアプローチは、デスティネーションデバイスがレガシーであり、各ソースデバイスがデスティネーションと通信するためにデスティネーションをモデル化しなければならない、という本発明の実施例で用いられる。これは、例えば、ファックスデバイスや電子メールデバイスの場合である。デバイスにレガシーネットワークと対話する能力を追加するのではなく、この機能はゲートウェイに置かれる。ゲートウェイは、通常、プログラム可能なプラットフォーム上にホスティングされる。

【0040】本発明のフレームワークでは、PnPプローカが、アクセスのためにゲートウェイを要求するレガシー（ダミー）デバイスとのセッションを確立したい場合、ゲートウェイは、PnPプローカから、そのデバイスの存在を隠蔽する。PnPプローカは、その代わりに、ゲートウェイとのセッションを作成し、ダミーデバイスのアドレスをゲートウェイに渡す。この時点以降、ゲートウェイは、セッションにおけるいずれかのチャネルを通じてデータを受け取ると、そのデータを、ゲートウェイ-ダミーリンクを通じてリモートデバイスへと中継する。本発明は、次の2つのタイプのゲートウェイを使用する。一般化ゲートウェイは、IPデバイスとの通信を行い、特殊化ゲートウェイは、ダミーを通じて非-IPデバイスとの通信を行う。特殊化ゲートウェイは、ダミーデバイスをネットワークに公開するものであり、そのデバイスを制御・管理するためのプログラミングロジックを含む。さらに、特殊化ゲートウェイは、ダミーデバイスのセキュリティポリシーを実現する。1つのゲートウェイデバイスが複数のダミーデバイスを管理することが可能である。

【0041】[PnPプローカ] PnPプローカは、アプリケーション、サービス、およびデバイスのサービスプローカとして作用する。PnPプローカは、分散システムのためのinfobusの概念を拡張し、ネットワーク全体をSoftware Busとして扱う。infobusシステムは当業者に周知である。詳細な説明は、Reaz Hoq

ue; "Connecting JavaBeans with InfoBus", Wiley Computer Publishing, Nov. 98. にある。Software Busもまた当業者に周知であり、Brian Oki, Manfred Pfleugl, Alex Siegel, and Dale Skeen, "The Information Bus - an architecture for extensible distributed systems", Proceedings of the 14<sup>th</sup> Symposium on Operating Systems Principles, p. 58-68, Asheville, North Carolina, December 1993. に記載されている。

【0042】PnPプローカにより、ネットワーク接続されたエンティティは、他のネットワーク接続エンティティ（ゲートウェイあるいはダミー）の能力を発見し利用することが可能となる。ゲートウェイは、その能力を、対応するPnPプローカに登録し、ダミーデバイスは、ゲートウェイを発見し、このゲートウェイを通じてPnPプローカと通信する。ネットワーク内のデバイスの各タイプ（例えば、X-10 X、1394 X、USB）ごとに特定のゲートウェイがあり、その一端でこのデバイスクラスタに接続され、他端でPnPプローカに接続される。X-10、HDMI、IEEE-1394、およびUSBは周知の通信プロトコルである。2つの相異なるゲートウェイが互いに対話したい場合、通信のためにPnPプローカを使用することができる。例えば、あるゲートウェイがJiniクラスタを代表する一方で、別のゲートウェイが、HDMIを用いたIEEE-1394クラスタのゲートウェイとなることが可能である。本発明のフレームワークでは、ゲートウェイは、そのアーキテクチャ用に設計されたダミーを使用することにより、受動（非TCP/IP）デバイスと通信する。

【0043】図2に、アクティブコンフィグレーションフレームワークの全体設計を示す。実行環境102内で動作するPnPプローカ104は、ゲートウェイ201を通じて、ダミーデバイス203およびユーザアプリケーション202と通信する。ゲートウェイデバイス201は、ネットワークノード205上で、そのオペレーティングシステム環境内で動作する。ノード205は、ゲートウェイおよびサービスへのアクセスを制御するためにセキュリティ実施機構204を提供する。

【0044】ネットワーク接続されたエンティティは、サービスユニットと呼ばれる有意味な機能に再分割される。原理的には、アーキテクチャは、ユーザアプリケーションの観点から、1つの有意味な機能を1つのサービスユニットとして定義する。サービスユニットは、ユーザアプリケーション/サービス全体であることも可能であり、ユーザアプリケーション/サービスの一部であることも可能である。コマンド、応答、およびデータは、ユーザアプリケーションとサービスユニットの間、サービスユニットとサービスの間、あるいはサービスユニットと別のサービスユニットの間で交換することが可能である。サービスユニットは、その機能をPnPプローカに登録する。

【0045】PnPプローカにより、ネットワーク接続されたエンティティは、他のネットワーク接続エンティティの能力を発見し利用することが可能となる。ネットワーク接続エンティティは、サービスユーザ（ユーザアプリケーション）であることが可能である。ユーザアプリケーションは、PnPプローカを通じて、サービスを発見し、それを使用することを要求する。PnPプローカは、サービスユニットとユーザアプリケーションを接続するためのトランスポート独立なインタフェース（PnPプローカインターフェースという）を提供する。このインターフェースは、TCP/IPプロトコルスタック上で動作するように設計される。PnPプローカは、サービスプローカまたはマネージャとしてその役割を果たすために、異なるクラスタ内の他のPnPプローカと通信する。PnPプローカ間のプロトコルはPnPプローカ間プロトコルと呼ばれる。

【0046】各デバイスクラスタは、高々、1つのPnPプローカを有する。ローカルなPnPプローカがないとき、ユーザアプリケーション／サービスユニットは、Java RMIやHTTPのようなインターネット通信プロトコルを通じてリモートPnPプローカ（別のゲートウェイに接続されたPnPプローカ）を使用することが可能である。Java RMIプロトコルは当業者に周知である。その詳細な説明は、Downing, Troy Bryan, 'Java RMI: Remote Method Invocation', IDG Books Worldwide, 1998、にある。HTTPプロトコルも当業者に周知である。その説明は、'Paul S. Hethmon, 'Illustrated Guide to HTTP', Manning Publications, 1997、にある。サービスユニットは、それに直接に接続されたPnPプローカを使用することも可能であるし、PnPプローカ間プロトコルを使用することによりリモートPnPプローカを使用することも可能である。図3に示すネットワークシステムでは、サービスユニット406は、PnPプローカ間プロトコル404を通じて、リモートPnPプローカ1104に接続されたサービス1406および1402を利用する。こうして、あるサービスユニットが、異なるデバイスクラスタに位置するサービスユニットで利用可能なサービスに問い合わせ、これを使用することができる。

【0047】[PnPプローカ機能コンポーネント] PnPプローカの内部コンポーネントを図4に示す。通常、PnPプローカは、サービスレジストリ305、サービスディスカバリ／アベイラビリティエージェント304、サービスセッション管理エージェント306、サービスロケーションプロトコル（SLP : service location protocol）ユーザエージェント308およびSLPサービスエージェント307からなる。PnPプローカ間プロトコル303は、他のPnPプローカ301との通信に使用される。PnPプローカインターフェース302は、ユーザアプリケーション103に対するインタ

フェースを提供する。サービスロケーションプロトコル（SLP）は、当業者に周知であり、James Kempf and Pete St. Pierre, 'Service Location Protocol for Enterprise Networks: Implementing and Deploying a Dynamic Service Resource Finder', John Wiley and Sons, 1999、に記載されている。サービスレジストリは、1個以上のサービスレコードを含む。サービスレジストリ内のすべての情報は、検索、索引づけおよび交換を容易にするために、XML (eXtensible Markup Language)として記述される。XMLもまた当業者に周知であり、Natanya Pitts-Moultis and Cheryl Kirk, 'XML black book', Coriolis Group Books, 1999、に記載されている。

【0048】[サービスレジストリ] PnPプローカは、サービスについての情報を保持するためにサービスレジストリを含む。最小限、レジストリは、そのPnPプローカに直接に接続されたゲートウェイおよびサービスについての情報を格納する。図3に示すように、これらのサービスは、ローカルPnPプローカ104またはリモートPnPプローカ1104にある。さらに、PnPプローカレジストリは、他のPnPプローカに登録されているサービスについての情報を格納することも可能である。この拡張されたレジストリ機能により、ローカルPnPプローカは、サービスのローカルなディレクトリを保持することが可能となる。これは、ローカルな環境にとって重要であり、ネットワーク内に別個の「集中化した」SLPディレクトリエージェントは不要となる。例えば、ローカルPnPプローカがプリントサーバをサポートする場合、レジストリは、すべての準拠プリントサーバについての情報を有することが可能である。最終的に、この情報は、負荷均衡、フォールトトレランス、あるいはキャッシングのために使用可能である。また、PnPプローカサービスレジストリは、デバイスタイルにかかわらず、ローカルPnPプローカの範囲内にあるすべての準拠デバイスについての情報を提供する、ネットワークディレクトリとして作用することも可能である。この場合、ネットワーク内のPnPプローカのうちの1つが、すべてのPnPプローカを見つけて登録する責任を負う中心ディレクトリとして指定されることになる。ネットワーキソースに対する他のデバイスからのすべての要求は、このPnPプローカへ向けて送られ、このPnPプローカはそれぞれに応答することになる。

【0049】[サービスレコード] サービスレコードは、ネットワーク接続されたクラスタ内の利用可能なサービスユニットのセットであり、利用可能なサービスと、他のPnPプローカから要求されたサービスとを記述する。サービスレコードは、そのローカルクラスタ内の0個以上のサービスユニットレコードからなる。サービスユニットレコードは、サービスユニットIDフィー

ルドと、それに続く0個以上の属性レコードからなる。サービスユニットIDフィールドは、サービスユニットのタイプと、そのサービスユニットによって提供されるサービスとを識別する。デバイスは、ただ1つのサービスユニットを有するが、複数の属性を有することも可能である。属性レコードは、属性IDフィールド、その値、および、アクセス制御情報からなる。属性レコードは、主に、それぞれのサービスまたはデバイスに対する細かいアクセス制御を実施するために使用される。さらに、属性レコードは、デバイスの現在の状態、および、それを変更するために使用可能なインターフェースの記述を格納する。

【0050】[サービスディスカバリ／アベイラビリティエージェント] PnPプローカは、リモートPnPプローカを発見し利用可能なサービスを識別するために、サービスディスカバリ／アベイラビリティエージェントを必要とする。サービスディスカバリ(発見)は、ローカルPnPプローカによって指定される、要求されるサービスタイプを、リモートPnPプローカ上で利用可能なサービスタイプと比較することによって実行される。ローカルPnPプローカからリモートPnPプローカへ要求サービスタイプを送信し、リモートPnPプローカからローカルPnPプローカへその応答を送信するために、上記のJava RMIやHTTPを使用することができる。要求サービスタイプの指定の検査により、PnPプローカは、リモートPnPプローカに登録されているすべてのあるいは特定のサービスの特性を決定することができる。サービスディスカバリ／アベイラビリティエージェントは、SLPユーザエージェントおよびSLPサービスエージェントの上に位置する。

【0051】さらに、PnPプローカは、サービスディスカバリ／アベイラビリティエージェントを用いて、サービスのアベイラビリティ(利用可能性)を周期的にチェックすることができる。ローカルPnPプローカは、適当なPnPプローカに対して、アベイラビリティチェックを実行するよう要求する。本発明の実施例では、ユーザは、アベイラビリティチェックの周期を指定し、また、このチェックを取り消すことができる。

【0052】[サービスセッション管理エージェント] ユーザアプリケーションがサービスまたはデバイスを見つけるときに、PnPプローカは、ユーザアプリケーションとそのサービスの間に仮想パイプ(トンネル)を確立する。これをサービスセッションという。このようなトンネルを用いて、ユーザアプリケーションとサービスの間で、コマンド、応答およびデータを交換することができる。デバイスのインターフェースの編成に従って、これらのコマンド、応答、およびデータは特定のフォーマットを有し、定義されたプロトコルのもとで交換される。PnPプローカは、この仮想パイプを管理しながら、以下の3つのプロトコルのうちの1つ

で動作するよう指令されることが可能である。

【0053】[ネイティブパケットでのネイティブデータ転送] PnPプローカは、データパイプを設定した後、バックグラウンドに入ることにより、ユーザアプリケーションとサービスが、メッセージストリームおよびデータフォーマットを管理することを可能にする。このアプローチは、PnPプローカが他のネットワークエンティティの能力を発見するために単独で使用され、アプリケーション、サービス、およびデバイスがユーザアプリケーションと発見されたサービスとの間の相互作用を管理するときに有用である。メッセージは、PnPプローカの関与なしに、ユーザアプリケーションとサービスの間で直接に交換される。メッセージ交換は、厳密に、ネイティブパケットにおけるネイティブデータの形式である。サービスを要求する前にサービスを発見するためあるいはサービスの問合せ(クエリ)を行うために、ユーザアプリケーションは、PnPプローカ間プロトコルを使用可能である。例えば、IEEE 1394対応のカメラは、このデータ交換フォーマットを用いて、データストリームをデジタルVCRに送ることができる。

【0054】[PnPプローカのパケットにおけるネイティブデータ転送(トンネリング)] データフォーマットはユーザアプリケーションおよびサービスによって選択され制御される一方で、PnPプローカが、データパイプを設定し、メッセージストリームを管理することも可能である。このアプローチは、ユーザアプリケーションと発見されたサービスとの間で共通のメッセージングプロトコルが存在しないときに有用である。すべてのメッセージは、PnPプローカ間プロトコルによって運ばれる。例えば、IEEE 1394対応カメラは、PnPプローカを通じて、異なるクラスタ内のデジタルVCRにデータストリームを送る。

【0055】[PnPプローカのパケットにおけるPnPプローカのデータ(完全機能ゲートウェイ)] PnPプローカが、データパイプを設定し、メッセージストリームを管理し、ユーザアプリケーションとサービスとの相互作用のためのデータフォーマット定義を提供する。また、プローカは、ユーザアプリケーションと発見されたサービスとの間の共通のメッセージングプロトコルおよび共通のデータフォーマットを提供する。メッセージ交換情報は、PnPプローカデータに含まれ、これはパケットにフォーマットされる。ユーザアプリケーションメッセージは、PnPプローカを通るが、PnPプローカは、メッセージの内容あるいはセマンティクスを検査することはない。このタイプの相互作用の例は、USBデバイスと対話するIEEE 1394対応デバイスである。

【0056】[SLPユーザエージェント] ユーザエージェントは、あるサービスとの接続を確立するためにユーザに代わって動作するプロセスである。ユーザエージ

エントは、サービスエージェントまたはディレクトリエージェントからサービス情報を取得する。

【0057】【SLPサービスエージェント】サービスエージェントは、サービスを公表するために、1個以上のサービスに代わって動作するプロセスである。

【0058】【プロトコルと相互作用】PnPプローカは、ユーザアプリケーションがネットワークのプラグアンドプレイ機能を利用することができるようにする標準化されたインターフェースを公開する。このPnPプローカインターフェースを用いて、アプリケーションは、トランスポート層とは独立にPnPプローカ間プロトコルを用いて相互に通信するように書くことができる。この設計は、ポートアビリティ（可搬性）を促進し、ネットワークサービスの発見および選択のためのスケーラブルなフレームワークを提供する。PnPプローカ間プロトコルは、IETF Service Location Protocol v.2に基づいており、サービスをサポートするネットワークホストの名前をユーザアプリケーションが知ることは不要である。むしろ、ユーザは、PnPプローカインターフェースを通じて、所望のサービスタイプと、対応する属性のセットとをPnPプローカに与える。これらは、PnPプローカに対して、所望のサービスを記述する。この記述に基づいて、PnPプローカは、PnPプローカ間プロトコルを用いてサービスのネットワークアドレスを解決し、PnPプローカ間プロトコルは、サービスロケーションプロトコルを使用する。また、PnPプローカインターフェースは、デバイスあるいはサービスのためのユーザの識別および認証機構を扱う。いったんサービスが識別されると、PnPプローカ間プロトコルは、Java RMIあるいはHTTPを用いて、ユーザアプリケーションが、発見されたサービスを使用することを可能にする。このフレームワークのもう1つの重要な特徴はゲートウェイによって提供される。ゲートウェイは、デバイスの能力をPnPプローカに登録し、PnPプローカが、PnPプローカにおいてサービスエージェントを通じてサービスを使用することを可能にする。

【0059】要約すれば、本発明のプロトコルスイートは、以下のプロトコルおよびインターフェースからなる。

- ・PnPプローカとユーザアプリケーションとの間のインターフェース。
- ・あるPnPプローカと別のPnPプローカとの間のプロトコル。
- ・PnPプローカとゲートウェイとの間のプロトコル。
- ・ゲートウェイとダミーとの間のプロトコル。

【0060】【ユーザアプリケーションへのPnPプローカのインターフェース（PnPプローカインターフェース】PnPプローカは、サービス登録、サービス発見、サービス要求、およびアベイラビリティチェックのために、ユーザアプリケーションに対して、標準化されたAPIおよびインターフェースを公開する。さらに、P

nPプローカは、適当なサービスあるいはデバイスのためのユーザの識別および認証を扱う。

【0061】【サービス登録】必要なとき、サービスユニットまたはユーザアプリケーションは、それぞれ、サービスユニットまたはユーザアプリケーションとしてローカルPnPプローカに自分を登録または登録解除するため、RegisterService()またはUnregisterService()を呼び出す。ユーザアプリケーションまたはサービスユニットがPnPプローカに登録された後、その機能は、別のユーザアプリケーションまたはサービスユニットからのQueryService()またはSearchService()要求に対する応答に含まれる。

【0062】【サービス発見】ユーザアプリケーションは、SearchService()を呼び出して、ローカルPnPプローカに対し、特定のサービスを有する登録されたサービスユニットを含むPnPプローカを探索するよう要求する。ユーザアプリケーションは、QueryService()を呼び出して、どのようなサービスユニットが登録されているか、および、ユーザアプリケーションによって指定されたPnPプローカに登録されているサービスユニットの機能を発見する。

【0063】【サービス要求】ユーザアプリケーションまたはサービスユニットは、OpenService()を呼び出して、ローカルPnPプローカに対し、ローカルPnPプローカまたはリモートPnPプローカのいずれかに登録されている特定のサービスユニットとのサービスセッションを開始するよう要求する。TransferData()、ReceiveData()、およびCloseService()コールにより、追加機能が提供される。

【0064】【アベイラビリティチェック】ユーザアプリケーションまたはサービスユニットは、StartAvailabilityCheck()を呼び出して、ローカルPnPプローカに対し、ローカルPnPプローカまたはリモートPnPプローカのいずれかに登録されている特定のサービスユニットが動作しているかどうかを周期的にチェックするよう要求する。

【0065】【ユーザの識別および認証】好ましい実施例では、本発明は、Javaアプリケーション環境を利用する。これは、分散コンピューティングのための適切な、現在利用可能なコンピューティングプラットフォームを提供するからである。この環境では、コードおよびデータの両方が、マシン間で移動可能である。この環境は、別のマシンからダウンロードされたコードが、妥当なレベルの信頼性で動作することを可能にする組込みセキュリティを有する。Javaアプリケーション環境における強い型付けにより、仮想マシン上で、オブジェクトのクラスの識別は、そのオブジェクトがそのマシン上で発生したものでないときでも実行可能である。その結果、ネットワークは、必要に応じて移動可能なオブジェクトの流動的なコンフィグレーションをサポートし、オ

ペレーションを実行するためにネットワークの任意の部分を呼び出すことができる。

【0066】ホームデバイスへの認証なしのアクセスは、フレームワーク全体のセキュリティにとって重大な結果を生じる可能性がある。リソースへの制御されたアクセスは、安全性とセキュリティの基礎である。従って、本発明の一実施例は、Java仮想マシンおよびJavaプログラミング環境を使用して、1つのデバイスのセキュリティを保証する。システムは、障害のあるデバイスに対処するように設計することが可能であるが、ネットワークセキュリティは、デバイスセキュリティよりも複雑な問題となる。

【0067】本発明のもう1つの実施例は、コードを参照する手段として、および、基本的なアクセス制御機構として、コードのMD5チェックサムを使用する。MD5は、コードのチェックサムを生成する暗号化手続きであり、アクセス制御のために使用される。しかし、ネットワークが大規模になると、セキュリティポリシーを指定しコードに名前付けするためより柔軟な分散機構が必要である。

【0068】本発明のセキュリティモデルは、プリンシバルとアクセス制御リストという2つの概念の上に構築される。サービスは、あるエンティティ(プリンシバル)に代わってアクセスされる。プリンシバルは、一般に、システムの特定のユーザに由来する。サービス自体が、そのサービスを実装するオブジェクトの識別に基づいて、他のサービスへのアクセスを要求することが可能である。サービスへのアクセスが許可されるか否かは、そのオブジェクトに対応するアクセス制御リストに依存する。

【0069】[PnPプローカ間通信プロトコル] PnPプローカ間通信プロトコルは、2つの部分に分かれます。第1に、ネットワーク内で新たなデバイスおよびサービスを発見するための、プロトコルおよび機構のセットが使用される。発見(ディスカバリ)手続きは、ユーザアプリケーションがどのようにしてネットワークインフラストラクチャを探査するか、および、ユーザアプリケーションがどのようにして自分自身およびそのサービスを登録することができるかを指定するサービスディスカバリプロトコルの部分に基づく。他方、ルックアップ手続きは、ユーザアプリケーションが、集中化されたレジストリがある場合あるいはない場合に、必要とするサービスを探査するためにどのようにしてレジストリに問い合わせを行うかを指定するプロトコルに基づく。

【0070】いったんサービスが探索された後、そのサービスを利用するため他のステップが必要になることがある。本発明の実施例では、ユーザアプリケーションは、サービスの品質およびセキュリティ条件を含めて、サービスへのアクセスの交渉を行う。さらに、サービスへのアクセスの交渉が成功した後、ユーザアプリケーションは、J ava R M I、HOP、あるいはHTTPプロトコルを使用して、実際にサービスを利用する。

【0071】[サービスディスカバリ/登録プロセス] ルックアップおよびディスカバリの両方のために、サービスロケーションプロトコル(SLP)が使用される。図5に、サービスディスカバリ手続きを示す。SLPは、自動的に、しかも事前の設定なしで、ユーザエージェント(UA:user agent)502の要求を、サービスエージェント(SA:Service Agent)504によって提供されるサービス512と照合する。SLPは、SAによるサービスの公表と、ディレクトリエージェント(DA:Directory Agent)511により管理されるSLPディレクトリ508へのサービスの編成とを処理する。また、SLPは、サービスがユーザアプリケーションにサービスの能力および設定条件を通知する手段を提供する。

【0072】PnPプローカ501内のUA502は、ユーザアプリケーション103によるサービスおよびリソースの要求を理解する。同じくPnPプローカ503(同じPnPプローカでも異なるPnPプローカでもよい)にあるSA504は、各ネットワークサービスを代表し、UA502にとって利用可能なサービスを行う。SLPは、動的にサービス属性を保持するため、UA502は、現在の情報を取得することが可能である。サービスは、アプリケーションによって自動的に探索されることも可能であり、あるいは、サービスブラウザでユーザに提示されることも可能である。SLPは、既存のアプリケーションをサポートするとともに、新たなサービスの公表および発見を容易に可能にする。

【0073】本発明によれば、サービスは、ゲートウェイによって記述される。ゲートウェイは、SA内の属性(そのサービスに対応する)の値を設定する。例えば、プリンタは、ポストスクリプトプリンタとして、青色の紙が利用可能なプリンタとして、あるいは、ユーザのオフィスの同じフロアにあるプリンタとして、記述することができる。UA502は、DA511への要求メッセージSrvReq505において必要なキーワードおよび属性値を指定することによって適当なプリンタを選択し、応答を待つ。DA511は、適当するサービスのネットワークロケーションを含む応答SrvRp1y505により応答する。小規模な施設では、DAがないことも可能であり、その場合、要求メッセージは直接にSAに送られることになる。これを、ブロードキャストによるディスカバリ(5.0.7)という。

【0074】サービスは、DA511に登録することによって自分自身を公表する。サービスの登録は、サービスを記述するすべてのキーワードと属性値対(属性と値からなる対)とからなるリストを含む。また、登録は、リソースへのリースを含む。これにより、リースの満期後、サービス情報はDA511によって削除される。リ

ース機構は、サービスが永久に公表され続けているのにサービスハードウェアがもはや利用可能でない状況を避けるために実装される。また、明示的な登録解除も、SAの規則正しいシャットダウン手続きの一部として、DAからサービスの情報を削除することが可能である。また、SAが、現在の属性情報を周期的に登録することにより、UAが、サービスに関連するステータス、負荷、温度、あるいはその他の動的特性を確認することも可能である。

【0075】本発明のフレームワークでは、サービスは、そのサービスタイプに従って分類される。各サービスタイプは、SA504がサービスを記述するために利用可能にする、利用可能なキーワードおよび属性のセットを定義する。サービスブラウザは、まず、UA502にとって利用可能なサービスタイプを発見した後、そのサービスタイプについてSAによって公表されているすべての情報を問い合わせることによって、UA502にとって利用可能なすべてのサービスのすべての特性を決定することができる。

【0076】[ユーザエージェントとサービスエージェントの相互作用] サービス/デバイス登録プロセスの一部として、まず、サービスユニット512のゲートウェイは、サービスあるいはデバイスをSA504に登録し、その後、SA504は、SrvReqメッセージ506をDA511に送る。この登録は、サービスの特定のインスタンスに対するサービスURIと、そのサービスを記述する属性値対とを含む。DA511は、さまざまな目的で、このような登録をキャッシュすることができる。登録がキャッシュされた後、DA511は、SrvAckメッセージにより応答する。また、サービス登録は、寿命(lifetime)を含む。サービスが利用不能になったがそれ自身を登録解除することができなかつた場合、寿命値により、DA511は、キャッシュされた登録を満期にすることができる。この状況は、SA504がSrvDeregメッセージ506を発行することができない場合にしか生じないはずである。正常動作中、SA504は、後続のSrvReqメッセージで周期的にサービスの登録をリフレッシュする。このような「リフレッシュ」メッセージは、いずれかの値が変化した場合には更新された情報を含むことが可能であるが、完全な属性のセットを含む必要はない。

【0077】PnPプローカ501のUA502は、サービスが要求されるときに、DA511に対して要求を行う。UA502がDA511を発見することができるいくつかの方法がある。起動時にDA511のアドレスをUA502に静的に設定することに加えて、UA502は、当業者に周知のDHCP(Dynamic Host Configuration Protocol)を用いてDA511のアドレスを要求することも可能である。第3のオプションとして、リンク上にマルチキャストを行い、DAの公表を受け取ると

いうことがある。これらの3つのオプションは共存するが、SLPが、手動設定なしで動作可能であることが重要である。この理由で、UA502は、DA511を見つけるためにSrvReqを使用することが可能である。SrvReqは、SLPに対するIANA割当てマルチキャストアドレスにマルチキャストされる。IANAは、当業者に周知のInternet Assigned Numbering Authority標準である。これはマルチキャスト要求であるため、複数のユニキャスト応答を受け取る可能性がある。その結果として得られるDAのリストは、他のサービス要求のために使用可能である。

【0078】いったんUA502がDA511のアドレスを得ると、後続のサービス要求は直接にそのエンティティに対して行うことが可能である。プリンタを例とすると、UA502がプリンタサービスを探索しようとする場合、SrvReqが作成される。このメッセージは、「サービスタイプ printer」と、オプションの属性および値のリストとを含む。属性値対は、要求されるプリンタのタイプを記述する。このメッセージは、事前に設定された、または、マルチキャストを通じて発見された、DA511へユニキャストされる。DA511は、SrvReqを受信し、キャッシュされた登録に対するルックアップを実行して、要求された属性および値と一致するものを見つけようと試みる。その後、SrvReplyが、要求側のUA502へユニキャストされる。この応答は、一致結果に依存して、0個以上のサービスURIを含む。その後、ユーザアプリケーションは、そのサービスURIを用いてプリンタを見つける。

【0079】施設内にDA511がない場合、PnPプローカは、同じくラスタ内のSA504を見つけることに制限される。これは、多くの小規模な施設では許容できるかもしれないが、PnPプローカのスケーラブルなアプリケーションの場合、および、複数のクラスタを有する大規模な施設では、DA511を使用しなければならない。さらに、SLP DA511は、LDAPベースのディレクトリ509へのゲートウェイであることが可能であるため、PnPプローカインターフェースは、変換スキーマ510を用いて、すべてのプロトコルに対する单一のAPIを提供する。

【0080】UA502は、サービスハンドルを取得したいとき、サービスタイプと、要求する属性の文字列ベースの問合せとをサービス要求として送信する。サービス応答が返されるとき、これは、サービスを指すURIを含む。URIは、SA504のIPアドレスを含むか、さもなければ、DNSがIPアドレスへと解決することができる名前を含む。また、URIは、UA502がサービスを使用するために必要とするその他の情報を含む。例えばプリンタの場合、キューネ名と、プリントサービスをホスティングしているコンピュータのアドレスとが返される。URIは、リソースロケーションを表す

ための従来の標準と同様の自然な方法で、サービスを見つけるために使用される。さらに、文字セット問題は、標準的な方法で対処することができる。

【0081】 [PnPプローカとユーザエージェントおよびサービスエージェントとの相互作用] SLPを使用することにより、PnPプローカは、同じタイプの多くの他の公表されたデバイスから適当なサービスの正確な選択をすることができる。これは、UA502によって要求されたキーワードおよび要求された属性値に一致するサービスのみを要求することによって行われる。この 10 ようなキーワードおよび属性値は、AND、OR演算子、一般的な比較演算子により、また、部分文字列マッチングを使用することにより、ブール式へと結合することができます。

【0082】 PnPプローカは、ユーザへのコンフィグレーション情報の提供を透過的にし、新たなサービスの設定の作業を容易にする。これらはいずれも、システムアドミニストレータを必要とする。SA504が近隣のDA511にサービスを公表するように設定された後、UA502は、さまざまなサービスがオペレーションを開始および停止するとともにネットワーク上で変化する条件に、動的に適応することができる。例えば、ウェブアプリケーションは、現時点でたまたま動作中であるシステムとは独立に、適当なプリンタが利用可能である限りそれを見つけることができる。

【0083】 ゲートウェイは、SA504に新たなデバイスあるいはサービスを追加したいとき、そのサービスの利用可能な属性およびキーワードを供給する。SA504は、プログラム的にSLPに登録を行い、サービスの固有のコンフィグレーション情報に基づいて属性に対する値を割り当てることができる。その後、SLPは、登録(公表)を処理し、ユーザアプリケーションとサービスの間のコネクションの確立を可能にする。

【0084】 新たなサービスタイプを必要に応じて定義することができる。実験的なサービスタイプを限定された用途で配備し、その後に一般用に公開することも可能である。これは、SLPのネーミング権限機能を用いて実行される。ほとんどの一般的なサービスタイプは、デフォルトで、IANA(Internet Assigned Numbering Authority)により標準化されたサービステンプレート(スキーム)定義を使用する。代替サービスタイプを定義することは、任意の代替ネーミング権限に知られている名前を使用するようにディレクトリエージェントを設定するのと同様に容易である。例えば、家庭でセキュリティサービスを提供するためには、「service:secure:Door」というサービスタイプを定義することができる。もちろん、このタイプのサービスハンドルを利用するユーザアプリケーションは、「secure」サービスと通信するために、そのサービスのネットワークプロトコルを使用しな 40

ければならず、さらに、この特別のサービスタイプのURIに含まれる情報を解析することができなければならない。これは、任意のユーザアプリケーション/サーバプロトコルの自然な動作に本来のことである。

【0085】 [SLPをLDAPとともに使用すること] LDAPv3(Lightweight Directory Access Protocol)は、当業者に周知であり、汎用のディレクトリアクセスプロトコルとして普及しつつある。LDAPはその名前にLightweight(軽量)とあるが、SLPはLDAPよりもさらに「軽量」である。また、SLPは、自動ディレクトリ管理を提供するため、および、階層的で自由度の少ない名前空間における不適切なリソース配置を必要としないために、リソース管理においてLDAPよりすぐれている。SLPにおいて新たなサービスタイプを追加することはLDAPの場合よりもずっと容易である。タイプによる問合せは、SLPの場合のほうがLDAPの場合よりも効率的である。既存のリソースに対する属性記述は、SLPでは利用可能であるが、LDAPでは不可能である。

【0086】 SLPでは、事前コンフィグレーションなしでのディスカバリが可能である。これに対して、LDAPは、はじめに、ディレクトリのアドレスと、LDAPが使用するディレクトリスキームの知識を必要とする。SLPでは、非標準的な属性と、標準化されたサービスタイプテンプレートの新しいバージョンが可能であるため、進化することができる。

【0087】 本発明の実施例は、SLPを使用して、LDAPディレクトリの管理を簡単化し、SLPが、必要に応じてLDAPディレクトリから取得される情報を返すことを可能にする。別の実施例では、SLPは、LDAPにサービスエントリを入力するために動的に使用され、SLP問合せはLDAP問合せにマッピングされて、LDAPがSLPのバックエンドとなる。このようなコンフィグレーションの1つの利点は、ユーザアプリケーションが、LDAPディレクトリから直接にユーザ認証情報を得ることである。

【0088】 [サービス利用プロセス] サービスディスカバリ/登録プロセスは、サービスロケーションプロトコルによって提供されるURIによって検索可能なサービスレコードを提供する。PnPプローカ間の交換能力は、サービスレコードを含むQueryService()メッセージを交換することにより提供される。この問合せ(query)は、ユーザアプリケーションが利用したいサービスの要件を記述し、リモートPnPプローカに対してサービスの詳細を要求する。

【0089】 受信側PnPプローカは、QueryService()メッセージを受信すると、受信したサービスレコードを自己のサービスレコードと比較する。その後、受信側PnPプローカは、一致したサービスレコードを含むサービスレコードを要求側PnPプローカに返す。このアプ

ローチは、すべてのサービスユニット、特定のタイプのサービスユニット、あるいは、特定の能力のセットを有するサービスユニットのうちで、ユーザアプリケーションが探索を行う技術を提供する。ユーザアプリケーションは、サービスレコードを受け取った後、サービスあるいはデバイスを使用するためにPnPプローカとの通信を開始することができる。

【0090】ローカルPnPプローカは、リモートPnPプローカへメッセージを送ることによって、ユーザアプリケーションとサービスの間のサービスセッションの確立を要求する。このメッセージは、ユーザアプリケーションサービスハンドル、リモートサービスユニットハンドル、プロトコルID（ネイティブ／トンネル／ゲートウェイ経由）、要求側（リクエスタ）ID、および要求側資格証明（credential）からなる。

【0091】リモートPnPプローカは、要求されたサービスが受容されるかそれとも拒否されるかを指定するリザルトコードで応答する。リモートPnPプローカは、ある時間後のリダイレクトまたはリトライを含むことも可能である。サービスが受容される場合、リモートPnPプローカは、サービスセッションの開始を識別するサービスハンドルを送信する。サービスセッションは、ユーザアプリケーションとサービスユニットの間、または、2つのサービスユニットの間でのデータ転送を扱う。ユーザアプリケーションは、サービスの使用を完了すると、ローカルPnPプローカに対して、「close service」（サービス終了）メッセージをリモートPnPプローカへ送るよう要求する。さらに、ユーザアプリケーションは、サービスが利用可能であるかどうかを周期的に判定する必要があるとき、それぞれのPnPプローカに対して、フレームワーク内のPnPプローカ間でアベイラビリティチェックを実行するよう要求することができる。アベイラビリティチェックは、1つのデバイス全体に対しても、あるいは、デバイス内で提供される特定のサービスに対しても、実行可能である。

【0092】[PnPプローカ／ゲートウェイプロトコル] PnPプローカ／ゲートウェイプロトコルは、簡単なメッセージ交換機構に基づいている。ゲートウェイは、既に説明したように、ゲートウェイに直接に接続されたサービスユニットを代表し、提供されるサービスを登録する。また、ゲートウェイは、PnPプローカにあるサービスエージェントに対する、任意のサービスのコンフィグレーションにおいてなされる変更を反映する。この登録は、リースに基づいており、ゲートウェイまたはサービスエージェントのいずれかにより更新可能である。登録は、サービスエージェントに現在の情報を提供する。PnPプローカは、特定のサービスに対する要求を受け取ると、まず、サービスエージェントをチェックして、必要なデバイスあるいはサービスをいずれかの直接接続されたゲートウェイが提供するかどうかを判定す

る。また、ゲートウェイにより、サービスエージェントは、ゲートウェイ内の特定のイベントに関心があることを登録し、そのイベントの発生の通知を受け取ることが可能である。このようなイベントは、ネットワークにおけるデバイス／サービスの追加／削除であることが可能である。サービスのサービスレコードがアプリケーションの要件に一致した場合、ユーザアプリケーションは、ゲートウェイ内のサービスユニットと、ユーザアプリケーションとの間のサービスセッションを設定することができる。

【0093】【イベント通知機構】本発明のフレームワークでは、ユーザアプリケーションは、イベント通知要求をサーバに登録することができるため、サーバは、新たなデバイスがオンラインになったとき、あるいは、デバイスの属性が変化したときに、ユーザアプリケーションに通知することになる。これを達成するため、本発明は、登録ユーザにイベント通知を配信するための機構を提供する。

【0094】ネットワークイベント通知プロトコルには、CORBA (Common Request Broker Architecture) イベントサービス、X-Windowsシステムイベント、SGAP、BSCWなどのいくつかの周知のものがある。例えば、CORBAイベントサービスは、Robert Orfali and Dan Harkey, 'Client/Server Programming with Java and CORBA', Wiley, 1996. に詳細に記載されている。

【0095】しかし、上記のイベント通知サービスは、特定のアーキテクチャで動作するように設計され、大規模なコードベースを課すため、軽量通知サービスにおいて実際に使用するのは困難である。

【0096】いくつかのタイプのイベントの通知のための登録に関して提供される能力に加えて、ユーザは、属性を通知要求に関連づけることができる。好ましくは、このような通知は、高い信頼性で配信され、完全に規則正しいエンドツーエンド配信を保証するべきである。しかし、通知が、信頼できないトランスポートを通じて配信されるとき、確認応答やリトライは不要である。ユーザアプリケーションは、イベントの需要側が、受信の明示的な確認を供給側に提供することを要求することも可能である。認証のために、ユーザは、通知にデジタル証明して、通知の正当性および完全性を保証することができる。

【0097】本発明のフレームワークは、SLPの使用により、2つの方法でイベント通知を達成する。第1実施例によれば、サービスエージェントは、サービスが更新された後、マルチキャストアナウンス（SrvReg）を送信する。更新を受信した後、ユーザエージェントは、ブラウザ更新を行うことが可能であり、DAは登録を更新することができる。このような使用法は、SLP使用では明示的に禁じられていないが、フラッディン

グ(flooding)を引き起こす可能性がある。さらに、このアプローチは、多数のサービス、異なる言語、およびデジタル署名に関して効率的に動作しない。また、SrvRegは、ネットワークにおいて追加／削除／変更された各サービスのURIおよび属性を含む。

【009.8】本発明のもう1つの実施例では、サービスエージェントは、ネットワーク内のすべてのユーザエージェントへのブロードキャストにより、SAAdvertsメッセージをアナウンスする。SAAdvertsメッセージは、提供されるすべてのサービスタイプリストを含む。SAAdvertsを得た後、ユーザエージェントは、送信側サービスエージェントへ適当なサービス要求をユニキャストする。サービスエージェントからのアナウンスは、指数バックオフ方式で行われる。

【009.9】[ゲートウェイ／ダミープロトコル] ゲートウェイ／ダミープロトコルは、特殊なプロトコルであり、ネットワークにより統合されるデバイスのタイプを反映する。ネットワークは、X-10、USBおよびIEEE1394のデバイス／サービスをそれぞれネットワークの残りの部分に接続するX-10ゲートウェイ、USBゲートウェイ、およびIEEE1394ゲートウェイからなることが可能である。ダミーデバイスは、サービスレコードに対応するタイプのレコードを独自(proprietary)フォーマットで格納する。ゲートウェイ／ダミープロトコルは独自のものであるが、PnPプローカに対しては、標準化されたインターフェースを公開する。

【010.0】[実施例] 本発明の一実施例によれば、ネットワークは、1個以上のPnPプローカを有する。PnPプローカは、ユーザといくつかのサービスとの間のインターフェースとして作用する。一実施例では、X-10デバイスに対するPnPプローカは、USBデバイスによって提供されるサービスにリンクされたいくつかのボタンあるいはその他のユーザインタフェースウィジェットを提供する。ここで用いた「ユーザインタフェース・ウィジェット」という用語の意味には、ユーザアプリケーションのグラフィカルユーザインタフェースを作成するために使用されるボタン、ダイアログ、テキストウィンドウ、スケールおよびその他のグラフィカルコンポーネントを含む。あるボタンがクリックされると、PnPプローカは、単に、X-10ゲートウェイを通じて、特定のサービスを呼び出す。以下のステップは、この実施例により、ユーザアプリケーションが、ネットワーク内の新たに登録されたサービスを利用するため必要とされる。

【010.1】1. 白熱電球が、家庭の電源アウトレットに挿入される。

【010.2】2. 白熱電球は、ダミーデバイスであるため、X-10デバイスを通じて電源線ネットワークに接続される。

【010.3】3. ホームネットワークでは、PnPプロ

50  
一力およびX-10ゲートウェイは、新たなデバイスの導入前に既に動作中である。

【010.4】4. X-10ゲートウェイは、利用可能なアドレスを周期的にポーリングし、いずれかのデバイスがアクティブになったかどうかをチェックする。

【010.5】5. ゲートウェイは、新たなデバイスがネットワークに入ったことを認識し、そのデバイスと、そのデバイスにより提供されるサービスとの両方を、PnPプローカのサービスエージェントに登録する。

【010.6】6. サービスエージェントは、デバイスを使用するために、あるリースを取得する。このリースは、各リース期間後に更新して、これにより、ゲートウェイに対して、サービスが依然として利用可能であるかどうかをチェックするよう要求しなければならない。

【010.7】7. サービスエージェントは、デバイスにより提供されるサービスを、サービスロケーションプロトコルを用いて、ディレクトリエージェントに登録する。

【010.8】8. ユーザは、デバイスを使用したい場合、PnPプローカインターフェースを用いて、要求するサービスに対する問合せおよび検索を行う。すると、PnPプローカにあるユーザエージェントは、ディレクトリエージェントと連絡をとるか、あるいは、(サービスエージェントおよびユーザエージェントがいずれも1つのPnPプローカ内にある場合には)直接にサービスエージェントへ行く。

【010.9】9. 新たに発見されたデバイスが、ユーザのデバイス記述と一致する場合、サービスエージェントあるいはディレクトリエージェントは、サービスパラメータとともに、固有のURIをユーザエージェントに返す。

【011.0】10. ユーザアプリケーションは、これが非TCP/IPデバイスであると判定し、これに従い、PnPプローカ内のサービスセッション管理エージェントは、ユーザアプリケーションとデバイスの間のセッションを確立する。しかし、このセッションでは、コマンド／データ転送は、PnPプローカをゲートウェイとして使用して、PnPプローカ間プロトコルにより行われる。

【011.1】11. また、ユーザアプリケーションにより提供されるインターフェースは、デバイス内の機能の実行に関する情報も含む。

【011.2】12. ゲートウェイは、サービスにおける状態の変化をサービスエージェントに知らせて更新する。

【011.3】図6において、本発明の一実施例による例示的なホームネットワークは、電源線(X-10)クラスタ613、電話線(HomePNA/xDSL)クラスタ602および606、CAT5(イーサネット(登録商標))クラスタ616、USBクラスタ610、な

らびに J i n i デバイスクラスタ 6 0 9などの、周知のネットワークコンポーネントにより構成される。レジデンシャル(住宅)ゲートウェイ 6 0 5は、イーサネット、Home PNA、xDSL、および J i n i を含むさまざまなネットワークを相互接続するために使用される。イーサネットクラスタは、ユーザのホーム PC 6 2 0を含む。ホーム PC 6 2 0は、イーサネットプロトコルにより、ディジタルカメラ 6 1 7およびウェブパネル 6 1 8のようなさまざまなデバイスに接続される。ウェブパネル 6 1 8は、NEC社の製品であり、タッチパッドを有し、ユーザに、インターネットに容易にアクセスする手段を提供するデバイスである。イーサネットクラスタ 6 1 6は、イーサネットハブ 6 1 9を通じてレジデンシャルゲートウェイ 6 0 5と通信する。イーサネットハブ 6 1 9もまた、インターネットへのホームネットワークのブリッジを提供する。また、ホーム PC 6 2 0は、USB配線を用いてスピーカ 6 1 1およびネットワークカメラ 6 1 2に、および、X-10配線を用いて白熱電球 6 1 4およびセキュリティマネージャ 6 1 5に接続される。さらに、ホームネットワークは、別のホーム PC 6 0 3および 6 0 7、テレビジョン 6 0 1、ならびに電話機 6 0 4を有する。ホーム PC 6 0 7を含む xDSL クラスタ 6 0 6は、ブリッジ 6 0 8を通じてレジデンシャルゲートウェイ 6 0 5に接続される。PC 6 2 0上で動作する PnP プローカは、ユーザが、ウェブブラウザに基づくインターフェース内で、新たなデバイス/サービスを検索することを可能にする。本実施例は、ユーザインターフェースとして、Microsoft Internet Explore<sup>r1</sup> v5.0 ウェブブラウザを使用するが、他の適当なブラウザも使用可能である。

【0114】デバイス/サービスは、対応するネットワークに挿入されると、設定可能なプロパティとともにブラウザに現れる。デバイス固有のゲートウェイは、デバイス/サービス記述を格納する。ディレクトリエージェントがネットワーク内にある場合、ディレクトリエージェントが、デバイス/サービスの機能の登録を扱う。正当な権限が与えられた後、ユーザは、デバイスのプロパティの変更や、デバイスの相互接続が可能である。PnP プローカは、デバイスに対するユーザの認証と、非互換の場合のデータ/メッセージの変換を扱う。例えば、X-10デバイス 6 1 4および 6 1 5は、PC 6 2 0上にあるゲートウェイデバイスを使用することによって、フレームワーク内の残りのデバイスと対話することができるダミーデバイスである。PnP プローカは、HTTP サーバへのアドオンとして実装されることも可能である。ユーザは、電源線デバイス 6 1 4および 6 1 5の状態(電灯のオン/オフ)を変更することができる。デバイス(例えば、カメラ 6 1 7)が TCP/IP ベースである場合、ユーザは、カメラにより提供されるインターフェースに基づいて手動設定を実行し、カメラにより撮ら

れる画像を見ることができる。ユーザは、ビデオオンデマンドサーバに対して、利用可能な映画のリストを問い合わせることができる。通常、(PC 6 0 3上のMPE G-2 デコーダを用いて)ネットワークに接続されたアナログ TV 6 0 1は、同じタイプのインターフェースを使用している場合には、ビデオオンデマンドサーバにも接続可能である。しかし、TV 6 0 1およびネットワークカメラ 6 1 2はインターフェースを共有していないため、ユーザは、カメラの画像を直接に TV で見ることはできない。J i n i デバイス(例えば、プリンタ)がネットワークに接続されている場合、このデバイスは、非 J i n i デバイスがあたかも J i n i デバイスであるかのようにして、非 J i n i デバイスとともに動作する。

【0115】[J i n i インタフェースの問題点に対する解決法] ネットワークプラグアンドプレイを達成する候補のうちの1つとしての J i n i インタフェースモデルの欠点については既に説明した。本発明は、J i n i セキュリティモデルを借りながら、上記の J i n i インタフェースの欠点を克服する。J i n i のインターフェースの問題点を解決するには、以下の4つのアプローチを使用可能である。第1に、標準に基づく相互運用性が提供可能である。このアプローチによれば、すべてのサービスは標準 API を有し、サービスは、それらの標準 API を実装することになる。第2のオプションは、サンドボックスに基づく相互運用性である。この場合、適当なセキュリティモデルを有する Java 仮想マシンが与えられれば、サービスは独立して動作することができる。第3のオプションは、リフレクションに基づく相互運用性である。この場合、アプリケーションは、サービスに対して、そのインターフェースについて質問し、リフレクション機構を通じて、このインターフェースの状態に影響を及ぼす。最後に、第4の方法は、実装に基づく相互運用性である。この場合、同じ人あるいは会社が、プロキシおよびサービスの両方を製造することになる。

【0116】J i n i とは異なり、コードをクライアントに移動しそれを Java 仮想環境内で実行する代わりに、本発明は、サービスのアドレスおよびそのサービスのインターフェースを記述するその他のパラメータを処理し、このインターフェース情報をクライアントに提供する。

【0117】本発明のフレームワークが提供する主な利点の1つは、利用可能な機能およびユーザ相互作用モードにおける変更に応じたエンドユーザアプリケーションの動的な適応および再構成である。一部のインターフェースモデルは、ユーザが「従来型」ソフトウェアとのみ相互作用できることに基づいている。このようなシステムでは、ユーザは、アプリケーションをカスタマイズする能力において制約があるため、アプリケーションは、デバイスの能力を最大限に、すなわち、サービス設計者が予測したユーザの需要あるいは提供したプロ

グラムインターフェースの限界まで、利用することができない。

【0·1·1·8】多くの場合、自由度はユーザインターフェースによって制限される。アプリケーションの状態およびアプリケーションの初期設定は、ユーザインターフェースを通じてのみ操作可能であるが、ユーザインターフェース自体はあまり設定の自由度はない。これは、モノリシックなプログラムにおいて顕著であり、アプリケーションの状態にフックを設けたり、適切に定義された外部プロトコルを通じてアプリケーションの状態にアクセス可能にしたりするのがきわめて不便である。これに対して、クライアント／サーバプログラムは、公開された相互作用プロトコルを提供するが、2つの大きな問題点を有する。インターフェースの仕様がしばしば臨時的(ad-hoc)であること、および、ネットワーク機能についてのアプリケーションのビューを(クライアントコードを修正することによって)変更することができるのプログラマのみであることである。

【0·1·1·9】上記のインターフェースの問題点を解決する1つのアプローチは、アプリケーションプログラムが、オンザフライでクライアントデバイスあるいはコンピュータに、例えばJavaアプレットとして、ダウンロードされることである。しかし、このアプローチの問題点として、エンドユーザが、関連するエンティティとして異種のサービスのセットとの相互作用のためにアプリケーションをカスタマイズすることができないことがある。すなわち、このアプローチでは、アプリケーションが透過的でないために、機能的に同一のサービスの場合でも、プロトコルにおける小さい相違を克服することができない。例えば、上記のインターフェースモデルに基づく電灯スイッチ制御プログラムは、新たな電灯スイッチに遭遇するたびに、それを使用するために異なるアプリケーションをダウンロードする必要がある。したがって、このアプローチは機能を公開するが、操作しやすい形式ではない。

【0·1·2·0】もう1つのアプローチは、さまざまなサービスの機能インターフェースを標準化し、アプリケーションがこれらのインターフェース標準をサポートすることを要求することにより、上記の問題点を回避することである。このアプローチの問題点は、多数のアプリケーション固有の標準を強制することが非実際的である点である。

【0·1·2·1】明らかに、上記のいずれとも異なるモデル、すなわち、インターフェースを公開する要求と、プロトコル標準について合意する要求とのバランスのとれたモデルが好ましい。この問題点を解決するための第1の困難は、サービスの利用可能な機能(インターフェース)を記述する単一ドキュメントスキーマを定義し、関連するプログラムおよびユーザインターフェースをサービスの集合に(およびその逆)関連づけることである。第2の

困難は、カスタムユーザインターフェースが利用可能でないときに、上記のスキーマで書かれたドキュメントを用いてユーザインターフェースを生成し、ランタイム環境を実装することができるソフトウェアを提供することである。

【0·1·2·2】本発明の一実施例は、コンポーネントベースのアプリケーションフレームワークを、コンポーネント記述のためのアーキテクチャ独立なドキュメントモデルとともに利用する。このようなコンポーネントベースのアプリケーションフレームワークの詳細な説明は、David Krieger and Richard Adler, "The emergence of distributed component platforms", IEEE Computer Magazine, p. 43-53, March 1998, にある。このフレームワークは、上記の2つの基本的なアプローチの特徴を組み合わせることにより、コードフラグメントのアップロード／ダウンロードを可能にし、アプリケーション固有でないインターフェースの記述および操作のために標準を課す。

【0·1·2·3】【本発明のアクティブコンフィグレーションモデル】本発明のアクティブコンフィグレーションモデルでは、XML (eXtensible Markup Language) 記述が、すべてのネットワークデバイスに関連づけられる。XMLが使用されるのは、XML記述が、(サーバ側での) デバイスの機能の公表として、静的で不变のインターフェース記述を提供するからである。さらに、このようなXMLサービスインターフェース記述を操作することによって、クライアントは、フレームワーク内のサービスを利用することができる。フレームワークは、操作のための標準的なロケーションを提供するために、それぞれのサービスオブジェクトにプログラムおよびユーザインターフェースを公開する。

【0·1·2·4】図7に、本発明のアクティブコンフィグレーションモデルを示す。本発明によれば、ネットワーク内のあらゆるデバイスあるいはサービス701は、その機能インターフェースの定義702を指定する。これらのインターフェース定義は、アナウンス705によってクライアントに伝えられる。これらのインターフェース定義は、サーバ側で静的(スタティック)である(CORBAにおけるIDLと同様)。これらのインターフェースは、ネットワーク内のすべてのエンティティ間で共有されるが、いずれのユーザアプリケーションによって操作されることも可能である。ユーザアプリケーションは、サービスインターフェースの使用、および、このようなインターフェースによって提示されるメタデータに対する完全なコントロールを有する。ユーザアプリケーション703によってインターフェースまたはその使用の状態に変更704があれば、リファレンス(参照)706によりデバイスあるいはサービスに反映される。

【0·1·2·5】したがって、本発明は、任意のネットワークサービスを構築するためのプログラム可能な基盤を提

供する。本発明の一実施例では、エンドユーザアプリケーションは、利用可能な機能および相互作用モードにおける変更に応答して動的に適応され再構成される。この実施例は、コードをダウンロードするJavaアプレットの考え方と、標準化されたインターフェースの記述および操作との利点を組み合わせている。本発明のフレームワークでは、ユーザ（あるいはマシンにより生成されたユーザアプリケーション）は、単に、サービスによって提供されるインターフェース記述を編集し、その編集をデバイスに反映させることによって、ネットワークあるいはデバイスの状態を変えることができる。さらに、デバイスあるいはサービスは、標準化されたAPIを通じてアクセス可能である。すべてのAPIを標準化しようとすることによって特徴づけられる従来のシステムとは異なり、本発明のフレームワークによれば、さまざまなネットワーキングデバイスのベンダは、自己の製品の記述をフレームワークにマッピングすることが可能である。こうして、ベンダは、構文的定義およびデバイスの能力に集中することができる。

【0126】本発明によれば、デバイス記述は、宣言的スタイルのXMLを用いて格納され、アプリケーションコードに追加して使用される。XMLデバイス記述の主な機能は、データおよび制御フロー情報を提供することである。この制御／データの分離を公開することにより、本発明は、メタデータを設計に明瞭に組み込むので、格納および操作がアプリケーションとは独立となるため、アプリケーションは、将来の変更が可能なように設計することができる。

【0127】本発明のフレームワークで用いられているような、プログラム／ユーザインタフェースを、それらが参照するオブジェクトから分離することは、当業者に周知のSmalltalk<sup>11</sup>のモデル／ビュー／コントローラ(M/V/C)アーキテクチャに類似している。モデル／ビュー／コントローラアーキテクチャについてさらに詳細には、G. Krasner and S. T. Pope, 'A Cook book for Using the Model View Controller User Interface Paradigm in Smalltalk-80', Journal of Object Oriented Programming, August/September 1988. に記載されている。

【0128】M/V/Cアーキテクチャでは、データ(モデル)は、データの表示(ビュー)およびデータを操作するイベント(コントローラ)から分離される。同様に、本発明のシステムにおける文書は、データを操作し見るためのユーザインタフェース／プログラムにそのデータを関連づける糊(グルー)として作用する。

【0129】本発明は、XML構文を利用して、デバイス記述スキーマを、XML文書型定義(DTD)として作成する。XMLは、構造化された情報を含む文書のためのマークアップ言語である。これは、SGMLのサブセットであり、階層的な名前付けされた値と、他の文書

を参照するための高度な手段の形で、自己記述型のカスタムマークアップを提供する。XSL(Extensible Style Sheets)やXLink(Extensible Linking Language)のような同類のプロトコルとともに、XMLは、付随する文書スキーマ(文書型定義:DTD)のグループを指定し、発見し、結合する能力を提供する。しかし、HTMLとは異なり、XMLにおけるタグのセットはフレキシブルであり、タグのセマンティクスは、その文書に付随するDTDによって定義される。Resource Description Format, Dublin CoreおよびXML-Datatypesのような他のメタデータマークアップ言語も提案されている。

【0130】本発明のフレームワーク内では、デバイスあるいはサービスは、自己の記述スキーマを、XML文書と、それに伴うDTDおよびスタイルシートとして作成する。このスキーマは、言語独立なサービス記述のため、ならびに、ユーザインタフェース(プログラム)をサービスオブジェクトに、およびその逆にマッピングするための、マークアップタグを提供する。また、スキーマは、サービスのインターフェースをXML/XSL定義内に組み込むので、バーサは、ユーザアプリケーションにコードをダウンロードすることを必要とせずに、これらのインターフェースを読み出すことができる。例えば、電灯スイッチのための<ON>タグは、デバイスがメソッド呼出しおよびその他のイベントをリスンするアドレスおよびポート番号を示すことが可能である。

【0131】ユーザインタフェースをこれらのサービス記述から生成するために、XMLバーサが、クライアントユーザアプリケーションによって使用される。クライアントは、語彙的な型をユーザインタフェースウィジットにマッピングし、XML/XSLを解析した後、ネットワークの現在のコンフィグレーションに対するユーザインタフェースを生成する。ユーザ(あるいはユーザアプリケーション)は、そのデバイスに対応する動的に生成されるUIウィジットを単にクリックすることによって、任意のネットワークデバイスの状態を変更することができる。このアクションは、ユーザのマシン上で現在のUI状態を変更するとともに、適当なコマンド

(デバイスベンダにより定義されたXML文書に埋め込まれる)を実行のためにデバイスに送る。本発明の実施例では、この目的のために、Java<sup>12</sup>で書かれたパブリックドメインのXMLバーサと、Internet Explorer<sup>13</sup>5.0のようなXML対応ウェブブラウザが使用される。

【0132】以上、本発明について、その好ましい実施例を用いて説明したが、当業者には容易に認識されるよう、本発明の技術的範囲および技術思想から離れることなく、形式および細部におけるさまざまな変更が可能である。

#### 【図面の簡単な説明】

【図1】アクティブコンフィグレーションフレームワークのインフラストラクチャの図である。

【図2】アクティブコンフィグレーションフレームワークの全体設計図である。

【図3】2つのプラグアンドプレイプローカ間の通信を示す図である。

【図4】プラグアンドプレイプローカの内部表現を示す図である。

【図5】サービス発見（ディスカバリ）手続きを示す図である。

【図6】アクティブコンフィグレーションフレームワークの実装例の図である。

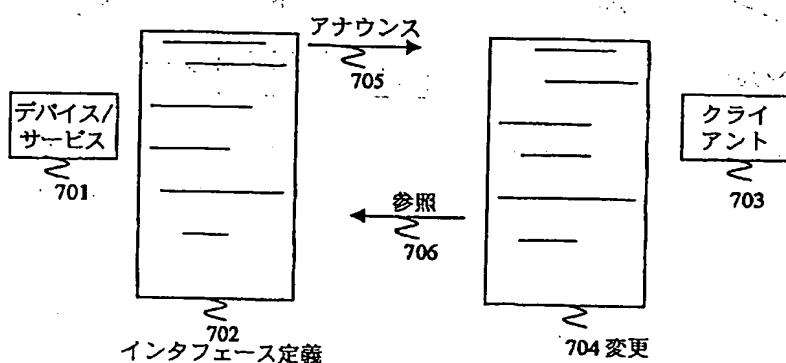
【図7】アクティブコンフィグレーションインターフェースモデルの図である。

#### 【符号の説明】

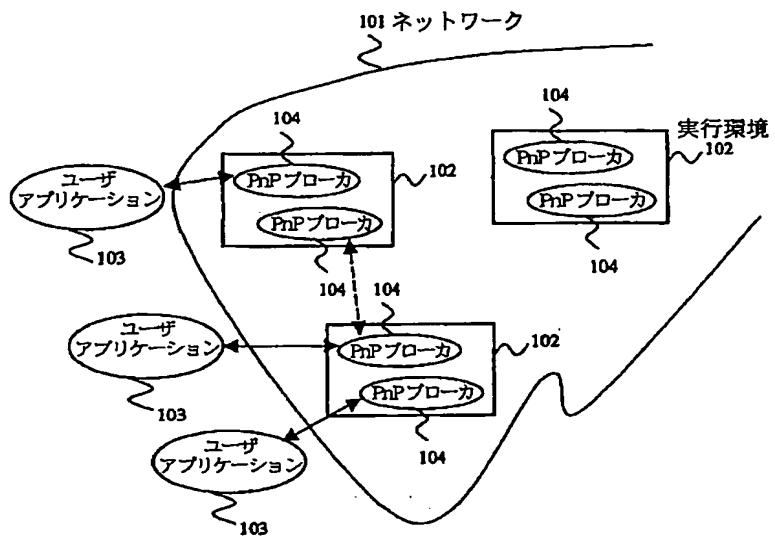
- 101 ネットワーク
- 102 実行環境
- 103 ユーザアプリケーション
- 104 PnPプローカ
- 201 ゲートウェイデバイス
- 202 ユーザアプリケーション
- 203 ダミーデバイス
- 204 セキュリティ実施機構
- 205 ネットワークノード
- 301 PnPプローカ
- 302 PnPプローカインターフェース
- 303 PnPプローカ間プロトコル
- 304 サービスディスカバリ／アベイラビリティエージェント
- 305 サービスレジストリ
- 306 サービスセッション管理エージェント
- 307 サービスロケーションプロトコル（SLP）サービスエージェント
- 308 SLPユーザエージェント
- 404 PnPプローカ間プロトコル
- 406 サービスユニット
- 501 PnPプローカ
- 502 ユーザエージェント（UA）

- 503 PnPプローカ
- 504 サービスエージェント（SA）
- 507 ブロードキャストによるディスカバリ
- 508 SLPディレクトリ
- 509 LDAPディレクトリ
- 510 変換スキーマ
- 511 ディレクトリエージェント（DA）
- 512 サービス
- 601 テレビジョン
- 10-602 Home PNAクラスタ
- 603 ホームPC
- 604 電話機
- 605 レジデンシャルゲートウェイ
- 606 xDSLクラスタ
- 607 ホームPC
- 608 ブリッジ
- 609 Jiniクラスタ
- 610 USBクラスタ
- 611 スピーカ
- 20 612 ネットワークカメラ
- 613 X-10クラスタ
- 614 白熱電球
- 615 セキュリティマネージャ
- 616 イーサネットクラスタ
- 617 デジタルカメラ
- 618 ウェブパネル
- 619 イーサネットハブ
- 620 ホームPC
- 701 デバイス／サービス
- 702 インタフェース定義
- 703 クライアント
- 704 変更
- 30 705 参照
- 706 リモートPnPプローカ
- 707 ユーザアプリケーション
- 708 変更
- 709 アナウンス
- 710 インタフェース定義
- 711 ユーザアプリケーション
- 712 変更
- 713 リモートPnPプローカ
- 714 ユーザアプリケーション
- 715 変更
- 716 リモートPnPプローカ
- 717 ユーザアプリケーション
- 718 変更
- 719 リモートPnPプローカ
- 720 ユーザアプリケーション
- 721 変更
- 722 リモートPnPプローカ
- 723 ユーザアプリケーション
- 724 変更
- 725 リモートPnPプローカ
- 726 ユーザアプリケーション
- 727 変更
- 728 リモートPnPプローカ
- 729 ユーザアプリケーション
- 730 変更
- 731 リモートPnPプローカ
- 732 ユーザアプリケーション
- 733 変更
- 734 リモートPnPプローカ
- 735 ユーザアプリケーション
- 736 変更
- 737 リモートPnPプローカ
- 738 ユーザアプリケーション
- 739 変更
- 740 リモートPnPプローカ
- 741 ユーザアプリケーション
- 742 変更
- 743 リモートPnPプローカ
- 744 ユーザアプリケーション
- 745 変更
- 746 リモートPnPプローカ
- 747 ユーザアプリケーション
- 748 変更
- 749 リモートPnPプローカ
- 750 ユーザアプリケーション
- 751 変更
- 752 リモートPnPプローカ
- 753 ユーザアプリケーション
- 754 変更
- 755 リモートPnPプローカ
- 756 ユーザアプリケーション
- 757 変更
- 758 リモートPnPプローカ
- 759 ユーザアプリケーション
- 760 変更
- 761 リモートPnPプローカ
- 762 ユーザアプリケーション
- 763 変更
- 764 リモートPnPプローカ
- 765 ユーザアプリケーション
- 766 変更
- 767 リモートPnPプローカ
- 768 ユーザアプリケーション
- 769 変更
- 770 リモートPnPプローカ
- 771 ユーザアプリケーション
- 772 変更
- 773 リモートPnPプローカ
- 774 ユーザアプリケーション
- 775 変更
- 776 リモートPnPプローカ
- 777 ユーザアプリケーション
- 778 変更
- 779 リモートPnPプローカ
- 780 ユーザアプリケーション
- 781 変更
- 782 リモートPnPプローカ
- 783 ユーザアプリケーション
- 784 変更
- 785 リモートPnPプローカ
- 786 ユーザアプリケーション
- 787 変更
- 788 リモートPnPプローカ
- 789 ユーザアプリケーション
- 790 変更
- 791 リモートPnPプローカ
- 792 ユーザアプリケーション
- 793 変更
- 794 リモートPnPプローカ
- 795 ユーザアプリケーション
- 796 変更
- 797 リモートPnPプローカ
- 798 ユーザアプリケーション
- 799 変更
- 800 リモートPnPプローカ
- 801 ユーザアプリケーション
- 802 変更
- 803 リモートPnPプローカ
- 804 ユーザアプリケーション
- 805 変更
- 806 リモートPnPプローカ
- 807 ユーザアプリケーション
- 808 変更
- 809 リモートPnPプローカ
- 810 ユーザアプリケーション
- 811 変更
- 812 リモートPnPプローカ
- 813 ユーザアプリケーション
- 814 変更
- 815 リモートPnPプローカ
- 816 ユーザアプリケーション
- 817 変更
- 818 リモートPnPプローカ
- 819 ユーザアプリケーション
- 820 変更
- 821 リモートPnPプローカ
- 822 ユーザアプリケーション
- 823 変更
- 824 リモートPnPプローカ
- 825 ユーザアプリケーション
- 826 変更
- 827 リモートPnPプローカ
- 828 ユーザアプリケーション
- 829 変更
- 830 リモートPnPプローカ
- 831 ユーザアプリケーション
- 832 変更
- 833 リモートPnPプローカ
- 834 ユーザアプリケーション
- 835 変更
- 836 リモートPnPプローカ
- 837 ユーザアプリケーション
- 838 変更
- 839 リモートPnPプローカ
- 840 ユーザアプリケーション
- 841 変更
- 842 リモートPnPプローカ
- 843 ユーザアプリケーション
- 844 変更
- 845 リモートPnPプローカ
- 846 ユーザアプリケーション
- 847 変更
- 848 リモートPnPプローカ
- 849 ユーザアプリケーション
- 850 変更
- 851 リモートPnPプローカ
- 852 ユーザアプリケーション
- 853 変更
- 854 リモートPnPプローカ
- 855 ユーザアプリケーション
- 856 変更
- 857 リモートPnPプローカ
- 858 ユーザアプリケーション
- 859 変更
- 860 リモートPnPプローカ
- 861 ユーザアプリケーション
- 862 変更
- 863 リモートPnPプローカ
- 864 ユーザアプリケーション
- 865 変更
- 866 リモートPnPプローカ
- 867 ユーザアプリケーション
- 868 変更
- 869 リモートPnPプローカ
- 870 ユーザアプリケーション
- 871 変更
- 872 リモートPnPプローカ
- 873 ユーザアプリケーション
- 874 変更
- 875 リモートPnPプローカ
- 876 ユーザアプリケーション
- 877 変更
- 878 リモートPnPプローカ
- 879 ユーザアプリケーション
- 880 変更
- 881 リモートPnPプローカ
- 882 ユーザアプリケーション
- 883 変更
- 884 リモートPnPプローカ
- 885 ユーザアプリケーション
- 886 変更
- 887 リモートPnPプローカ
- 888 ユーザアプリケーション
- 889 変更
- 890 リモートPnPプローカ
- 891 ユーザアプリケーション
- 892 変更
- 893 リモートPnPプローカ
- 894 ユーザアプリケーション
- 895 変更
- 896 リモートPnPプローカ
- 897 ユーザアプリケーション
- 898 変更
- 899 リモートPnPプローカ
- 900 ユーザアプリケーション
- 901 変更
- 902 リモートPnPプローカ
- 903 ユーザアプリケーション
- 904 変更
- 905 リモートPnPプローカ
- 906 ユーザアプリケーション
- 907 変更
- 908 リモートPnPプローカ
- 909 ユーザアプリケーション
- 910 変更
- 911 リモートPnPプローカ
- 912 ユーザアプリケーション
- 913 変更
- 914 リモートPnPプローカ
- 915 ユーザアプリケーション
- 916 変更
- 917 リモートPnPプローカ
- 918 ユーザアプリケーション
- 919 変更
- 920 リモートPnPプローカ
- 921 ユーザアプリケーション
- 922 変更
- 923 リモートPnPプローカ
- 924 ユーザアプリケーション
- 925 変更
- 926 リモートPnPプローカ
- 927 ユーザアプリケーション
- 928 変更
- 929 リモートPnPプローカ
- 930 ユーザアプリケーション
- 931 変更
- 932 リモートPnPプローカ
- 933 ユーザアプリケーション
- 934 変更
- 935 リモートPnPプローカ
- 936 ユーザアプリケーション
- 937 変更
- 938 リモートPnPプローカ
- 939 ユーザアプリケーション
- 940 変更
- 941 リモートPnPプローカ
- 942 ユーザアプリケーション
- 943 変更
- 944 リモートPnPプローカ
- 945 ユーザアプリケーション
- 946 変更
- 947 リモートPnPプローカ
- 948 ユーザアプリケーション
- 949 変更
- 950 リモートPnPプローカ
- 951 ユーザアプリケーション
- 952 変更
- 953 リモートPnPプローカ
- 954 ユーザアプリケーション
- 955 変更
- 956 リモートPnPプローカ
- 957 ユーザアプリケーション
- 958 変更
- 959 リモートPnPプローカ
- 960 ユーザアプリケーション
- 961 変更
- 962 リモートPnPプローカ
- 963 ユーザアプリケーション
- 964 変更
- 965 リモートPnPプローカ
- 966 ユーザアプリケーション
- 967 変更
- 968 リモートPnPプローカ
- 969 ユーザアプリケーション
- 970 変更
- 971 リモートPnPプローカ
- 972 ユーザアプリケーション
- 973 変更
- 974 リモートPnPプローカ
- 975 ユーザアプリケーション
- 976 変更
- 977 リモートPnPプローカ
- 978 ユーザアプリケーション
- 979 変更
- 980 リモートPnPプローカ
- 981 ユーザアプリケーション
- 982 変更
- 983 リモートPnPプローカ
- 984 ユーザアプリケーション
- 985 変更
- 986 リモートPnPプローカ
- 987 ユーザアプリケーション
- 988 変更
- 989 リモートPnPプローカ
- 990 ユーザアプリケーション
- 991 変更
- 992 リモートPnPプローカ
- 993 ユーザアプリケーション
- 994 変更
- 995 リモートPnPプローカ
- 996 ユーザアプリケーション
- 997 変更
- 998 リモートPnPプローカ
- 999 ユーザアプリケーション
- 9999 変更

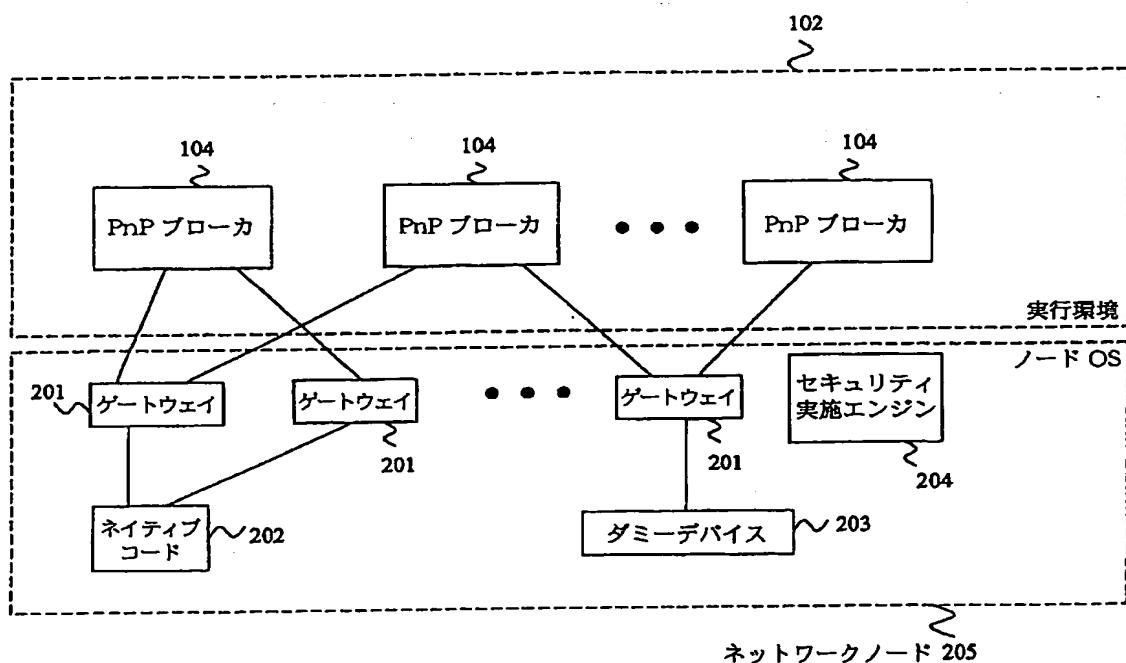
【図7】



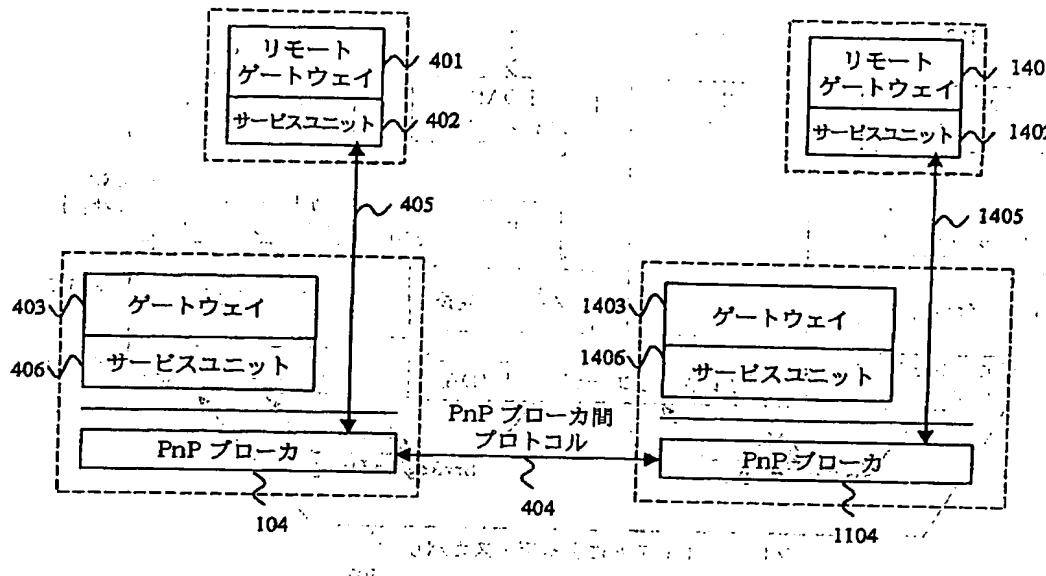
【図 1】



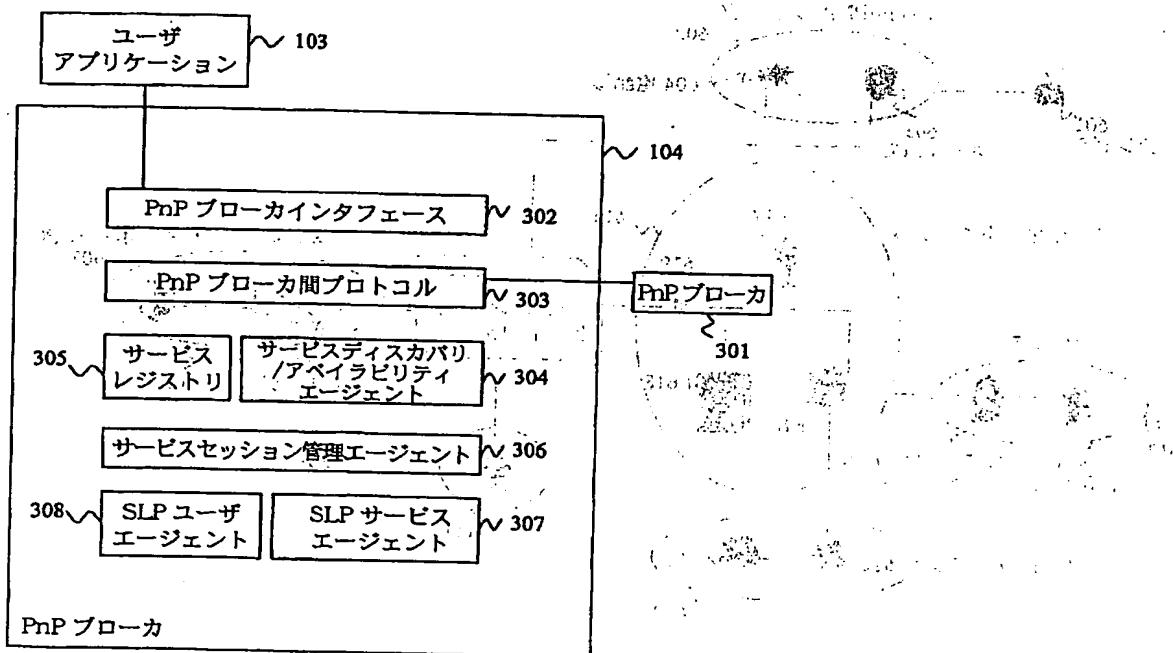
【図 2】



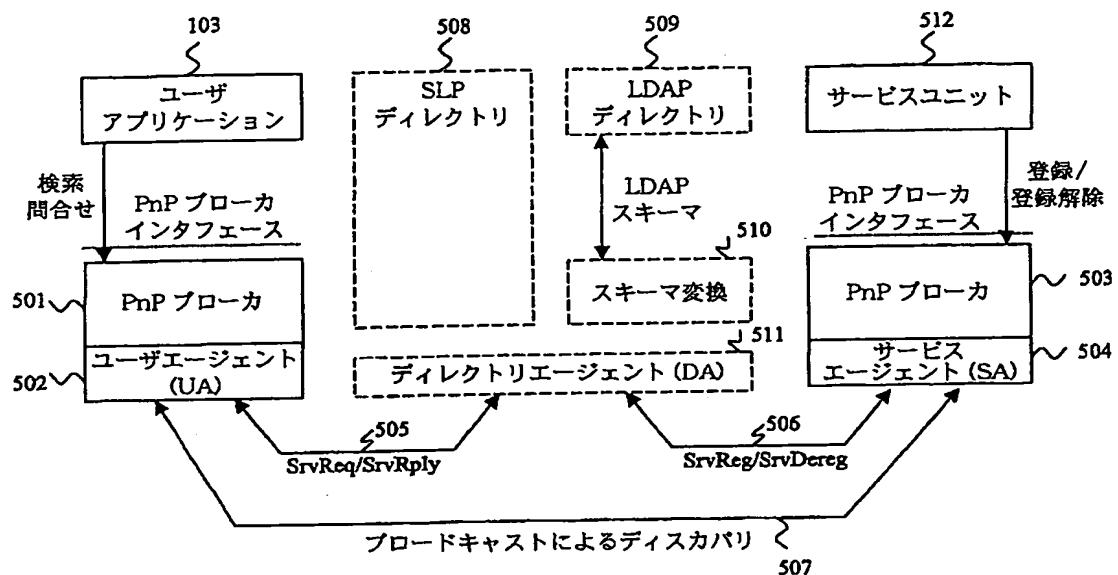
【図3】



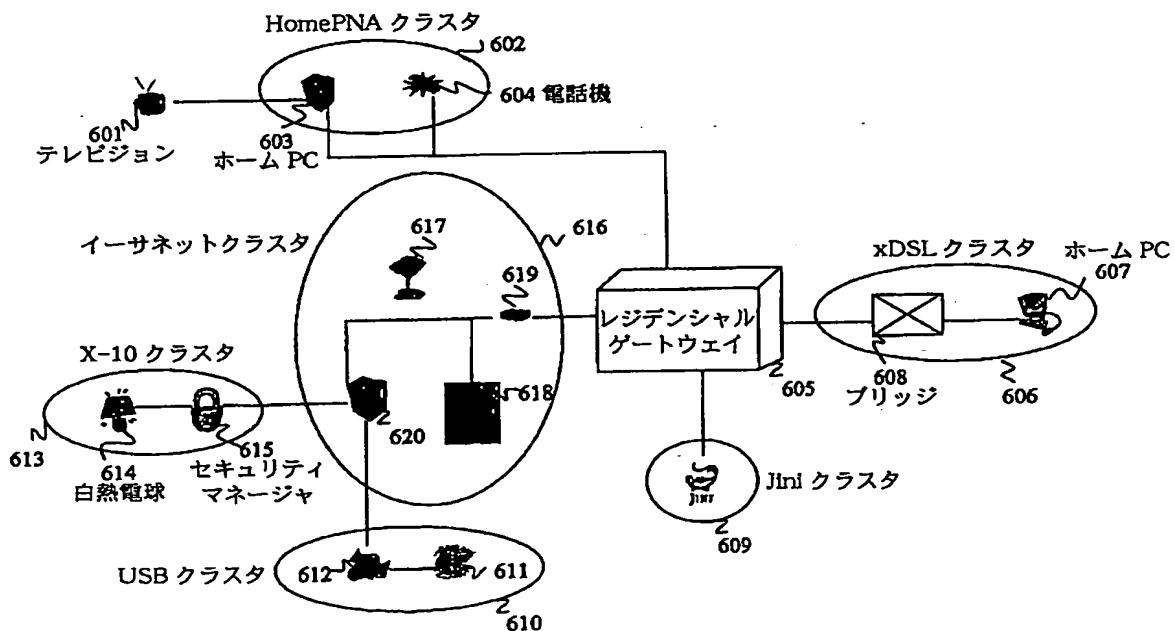
【図4】



【図5】



【図6】



フロントページの続き

(51) Int. Cl.?

H 04 L 29/06

識別記号

F I  
H 04 L 13/00

テーマコード(参考)

305 B